Rapport de Stage
DEA Logique et fondements de l'informatique 2001/2002

Vérification paramétrée de réseaux à processus probabilistes

Application du théorème de Hammersley et Clifford
aux algorithmes distribués

Stephane Messika
Direction de Claudine Picaronny et Laurent Fribourg

Introduction

L'objet de ce stage était de s'intéresser à des algorithmes distribués probabilistes. Il est frappant de remarquer la ressemblance entre l'évolution des machines de ces algorithmes et celle des états des spins (par exemple) en physique statistique.

Pourtant, les méthodes mathématiques (notamment probabilistes) utilisées pour traiter des équilibres de ces réseaux dans les deux domaines sont très différentes.

Nous nous sommes donc penchés sur les méthodes probabilistes de Physique statistique (recuit simulé, Monte Carlo,...) avant de penser à nous servir d'un théorème (Hammersley Clifford) permettant d'introduire une énergie caractérisant l'évolution du réseau de machines sous l'effet de l'algorithme.

Les résultats que nous obtenons ne sont pas nouveaux mais la méthode pour y parvenir semble être intéressante car elle est générale.

Nous avons en particulier traité d'un exemple précis (celui de l' algorithme de Herman) d'auto-stabilisation, mais la méthode se généralise à d'autres algorithmes.

Le mémoire est composé de trois parties, la première introduisant les algorithmes distribués et en décrivant certains ; la seconde permettant de situer la méthode, le théorème utilisé et ses inférences en Physique Statistique ; la dernière détaillant les résultats que nous avons obtenus.

Chapitre 1

Introduction aux algorithmes distribués

Le but de cette partie est de donner les définitions de base pour pouvoir travailler sur les algorithmes distribués [Lyn97].
Nous détaillerons aussi plus précisément les algorithmes traitant d'auto-stabilisation [Dij74] et en donnerons des exemples.

1.1 Définitions

Nous appellerons "machines", dans toute la suite, chacun des objets à qui s'appliquent l'algorithme.

Définition 1 (Algorithme distribué) *Un algorithme distribué est un algorithme censé fonctionner sur un nombre important de machines interconnectées. Chacune travaille de manière concurrente et indépendante, ce qui permet à l'algorithme de fonctionner correctement même si les machines avancent à des vitesses différentes.*

Définition 2 (Topologie) *La topologie d'un système est le graphe formé des machines pour sommets et des interconnections entre machines pour arêtes.*

1.2 Auto-Stabilisation

L'auto-stabilisation est une notion introduite en informatique par Dijkstra [Dij74], elle permet de décrire des algorithmes qui partant d'une configuration initiale quelconque se retrouve automatiquement dans une des configurations satisfaisantes (définies préalablement, leur ensemble sera noté $\mathcal{L}$ par la suite). Ces algorithmes ont éveillé l'intérêt des informaticiens, car ils

permettent de s'intéresser à des problèmes de tolérance aux fautes.
Dans notre cas, nous traiterons essentiellement de l'auto-stabilisation probabiliste qui garantit d'atteindre la classe $\mathcal{L}$ avec une probabilité 1 [Dol00].

En voici une définition plus formelle :

La définition suivante garantit qu'une fois qu'on a atteint la classe des états légitimes on n'en sort plus.

Définition 3 (Fermeture) *Une propriéte P est fermée sous l'éxécution du système S si une fois que P est vraie dans S il ne peut plus être faux.*

Par exemple une fois que l'algorithme atteint une classe $\mathcal{L}$ de configurations légitimes il n'en sort plus.

Puis maintenant on définit la notion de convergence qui assure l'accession à la classe des états légitimes. (Ici ceux qui vérifient une certaine propriéte P.)

Définition 4 (Convergence) *Un système est convergent si partant d'une configuration globale arbitraire, il atteint une configuration satisfaisant une propriété P avec un nombre fini de transitions.*

Enfin, on peut maintenant écrire la définition exacte de l'auto-stabilisation.

Définition 5 (Auto-stabilisation) *Un système est dit auto-stabilisant pour une propriété P s'il a les propriétés de fermeture et de convergence. Ici la propriété P sera "Appartenir à la classe $\mathcal{L}$".*

Dans la suite, nous parlerons essentiellement d'auto-stabilisation probabiliste [DFP01], dans le sens où les algorithmes ayant certaines transitions aléatoires, la convergence est obtenue avec une probabilité 1 ; nous parlerons aussi d'auto-stabilisation dans ce cas.

Une dernière définition est bien nécessaire à la compréhension de la suite, c'est celle du Scheduler :

Définition 6 ("Scheduler") *Le scheduler (parfois appelé Démon) est l'entité (réelle ou non) qui détermine dans quel ordre les machines sont sélectionnées dans l'algorithme.*

Illustrons ces quelques définitions par des exemples qui seront étudiés par la suite.

1.3 Exemples

Il existe de nombreux exemples d'algorithmes distribués et d'algorithmes d'auto-stabilisation (notamment ceux de Dijkstra), mais nous nous intéresserons ici seulement à ceux qui sont probabilistes. En effet le fait d'introduire des probabilités permet d'ajouter une disymétrie qui souvent débloque des problèmes. Par exemple Lehmann et Rabin [LR81] ont démontré qu'il n'existait pas de solutions déterministes au problème du "dîner des philosophes"(algorithme traitant du partage de ressources), mais en ont trouvé une en ajoutant des probabilités. Nous donnerons donc ici deux exemples d'algorithmes distribués d'auto-stabilisation probabiliste.

1.3.1 L'algorithme de Herman

Introduction

Cet algorithme est un algorithme d'auto-stabilisation sur un anneau (voir [Her90]. Le but est que partant d'une configuration quelconque sur les n machines (valant 0 ou 1) on obtienne une configuration avec un seul jeton (c'est à dire une et une seule suite de deux chiffres égaux consécutifs). Il est évident que pour cela le nombre de machines doit être impair (sinon la configuration 0101...0101 ne contiendrait aucun jeton). Ici donc, la classe $\mathcal{L}$ d'états légitimes est donc composée des configurations ne comprenant qu'un seul jeton. Avant de donner formellement l'algorithme expliquons-le par un exemple simple.

Imaginons 17 (ou n'importe quel nombre impair) personnes en cercle, et jouant à pile ou face. Ils jètent tous leur pièce (état initial), puis chacun regarde le résultat de son voisin de gauche ; s'il est différent du sien il retourne sa pièce, s'il est identique il la rejette. Puis ils regardent tous à nouveau leur voisin de gauche. Le résultat de Herman est que s'ils suivent cet algorithme, au bout d'un temps fini il n'y aura qu'une seule personne à la fois qui relancera sa pièce.

En voila la traduction formelle :

L'algorithme

Soit n le nombre impair de machines dans l'anneau, on notera x_i la valeur prise par la machine k ou k est égal à i modulo n. Enfin on notera γ_i la variable aléatoire prenant 0 comme valeur avec probabilité 1/2 et 1 avec probabilité 1/2.

L'algorithme est donc le suivant : (il est synchrone)

Pour tout i de 1 à n :

– Si $x_i = x_{i-1}$ alors $x'_i = \gamma_i$

– Si $x_i \neq x_{i-1}$ alors $x'_i = x_{i-1}$

1.3.2 L'algorithme de Flatebo et Datta

Introduction

L'objectif de cet algorithme (voir [FD94]) est exactement le même que celui de Herman. En effet notre objectif est toujours de réduire à 1 le nombre de jetons contenus dans un anneau. Cet algorithme est différent dans le sens où il nécessite deux machines exceptionnelles. Ces deux machines, en effet ne suivront pas le même algorithme que les autres.

L'algorithme

Le scheduler est soit probabiliste, on choisit de manière aléatoire quelle machine va agir, soit synchrone. Les deux machines exceptionnelles sont notées 0 et $N-1$

Pour 0 :

Si $x_0 = x_{N-1}$ alors $x'_0 = 1 - x_0$ il reste inchangé sinon.

Pour $N-1$:

Si $x_{N-1} \neq x_{N-2}$ alors $x'_{N-1} = 1 - x_{N-1}$ avec la probabilité p et ne change pas avec la probabilité $1-p$

Pour tous les autres machines :

Si $x_i \neq x_{i-1}$ alors $x'_i = x'_{i-1}$.

1.3.3 Conclusion et remarques

La preuve de l'auto-stabilisation de ces deux algorithmes a été faite, mais chacune d'elle est spécifique dans le sens où on se doit de trouver une mesure pour chacun des algorithmes qui décroit (et qui dépend du nombre de jetons) avec une probabilité positive étape après étape et qui nous permet de conclure quant à la convergence (la fermeture étant simple à démontrer).

Nous tenterons par la suite de donner une méthode plus générale pour démontrer ces résultats de convergence.

Chapitre 2

Le théorème de Hammersley et Clifford en physique

Ce chapitre est dédié à l'explication de la génèse des idées qui nous ont guidés. Nous nous sommes inspirés des notations et des résultats de [Br9] pour les parties mathématiques et de [PF91] pour la Physique statistique.

Notamment, nous parlerons des ressemblances entre la physique statistique et les algorithmes probabilistes distribués, et nous donnerons un théorème mathématique (Hammersley-Clifford) qui nous permettra de mener à bien nos calculs dans la partie suivante.

Tout d'abord voici deux exemples de domaines de Physique qui nous frappent par leurs points communs avec la théorie des algorithmes distribués.

2.1 Modèle d'Ising

Le modèle d'Ising a été créé en 1920 par Lenz pour expliquer le ferromagnétisme. Les atomes de fer sont situés aux sommets d'un réseau périodique, et chaque atome se comporte comme un petit aimant qui peut s'orienter selon une direction notée $+1$ ou selon son opposée noté -1. On appelle spin cette valeur et on la notera $\omega(i)$, l'énergie d'un couple d'atomes **voisins** est donnée par

$$H = -J\omega(i)\omega(j)$$

où J est une constante positive. Des considérations dues au principe de Carnot (Thermodynamique) nous permettent d'obtenir une probabilité sur l'ensemble des configurations ω du système donnée par

$$P(\omega) = \frac{1}{Z(J)}e^{-H(\omega)}$$

où $Z(J)$ est une constante normalisatrice appelée fonction de partition et où $H(\omega)$ est la somme sur tous les atomes voisins des énergies. On verra par la suite qu'un champ régi par une loi de probabilité de la sorte est appelé **Champ de Gibbs.**

2.2 Restauration Bayesienne d'images

Une image en deux dimensions peut être modélisée par une matrice de pixels donnant à chaque endroit le niveau de gris. Les propriétes d'une image sont telles que le niveau de gris d'un pixel donné ne dépend que de ceux de ses voisins (de manière probabiliste), il est, en effet très probable qu'un pixel, entouré de pixels tous blancs, soit lui-même blanc. Cette notion définie ici de manière informelle a une définition plus précise que nous allons développer tout de suite : celle de **Champ de Markov**

L'objectif de la suite de ce chapitre est d'expliquer les notions de Champs de Gibbs et de Markov, de donner le théorème établissant leur équivalence et d'en donner quelques applications.

Avant d'aller plus loin, il est donc nécessaire de donner quelques prompts rappels de probabilités.

2.3 Rappels de probabilités

Nous ne détaillerons pas ici les résultats classiques de théorie des probabilités, mais nous nous attacherons à expliciter les notions utiles à la compréhension des chaines et champs de Markov.

Définition 7 (Probabilité conditionnelle) *Soient A et B deux événements tels que $P(B) \neq 0$, on appelle probabilité conditionnelle de A sachant B la valeur*

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Définition 8 (Processus aléatoire) *Un processus aléatoire à valeurs dans E est un terme*

$$X = (\Omega, \mathcal{F}, (X_n), \mathbf{P})$$

où$(\Omega, \mathcal{F}, \mathbf{P})$ est un espace de probabilité et où X_n est une variable aléatoire à valeurs dans E.

On notera $\mathcal{F}_n$ la tribu engendrée par $X_0, X_1, ..., X_n$

Définissons à présent les chaînes de Markov. Les chaînes de Markov sont des objets cruciaux en probabilité : elles modélisent les processus ne se souvenant pas du passé lointant et évoluant donc en ne prenant en compte que

l'état courant du système.
Soit formellement :

Définition 9 (Chaîne de Markov) *Un processus aléatoire* $X = (\Omega, \mathcal{F}, (X_n), \mathbf{P})$ *est une chaîne de Markov ssi*

$$P(X_{n+1} \in A | \mathcal{F}_n) = P(X_{n+1} \in A | X_n).$$

2.4 Champs de Gibbs et champs de Markov aléatoires

Il est donc maintenant nécessaire d'étendre un peu le domaine classique de la théorie des probabilités, pour définir de nouvelles notions, un peu moins utilisées qui sont celles de champ de Markov et champ de Gibbs.

Définition 10 (Champs aléatoires) *Soit un* S *un ensemble fini d'éléments appelés sites et* Γ *un ensemble fini appelé l'espace de phase ; un champ aléatoire sur* S *avec des phases sur* Γ *est un ensemble de variables aléatoires* $X = (X(s))_{s \in S}$ *à valeurs dans* Γ

L'extension de la notion de passé immédiat valable pour les champs de Markov est ici caractérisée par une topologie et des voisinages dont voici les définitions :

Définition 11 (Voisinages, Topologie) *Une topologie sur* S *est une famille* $N = (\mathcal{N}_s)_{s \in S}$ *de voisinages tels que pour tout* s *de* S *on ait :*

1. $s \notin S$
2. $t \in \mathcal{N}_s \Rightarrow s \in \mathcal{N}_t$

Définition 12 (champ de Markov aléatoire) *Un champ aléatoire est appelé champ de Markov aléatoire (CMA) respectant la topologie* N *si pour tout les sites* s *dans* S *on a :*

$$P(X(s) = x(s) | X(S \backslash s) = x(S \backslash s)) = P(X(s) = x(s) | X(\mathcal{N}_s) = x(\mathcal{N}_s)) \tag{2.1}$$

Cette définition est clairement une extension des chaines de Markov a des espaces de dimension plus grande que 1. En effet sachant la configuration générale, un état ne dépend que de ces voisins ; de plus une chaîne de Markov est un champ aléatoire de Markov où S est $\mathbb{N}$ et le voisinage est réduit à l'entier suivant et au précédent.

Définition 13 (Spécification locale) *La caractérisation locale du CMA au site s est donnée par la fonction :* $\pi^s : \Gamma^S \to [0,1]$ *définie par*

$$\pi^s(x) = P(X(s) = x(s) | X(\mathcal{N}_s) = x(\mathcal{N}_s)) \tag{2.2}$$

Nous allons, à présent, définir la notion de Champs de Gibbs (qui s'avérera être équivalente à celle de champs de Markov) et qui est très connue en Physique statistique, il est nécessaire de donner quelques définitions au préalable :

Définition 14 (clique) *Un singleton s est une clique. Un sous-ensemble $C \subset S$ avec plus d'un élément est une clique si et seulement si deux sites distincts de C sont des voisins mutuels.*

Moins formellement, une clique est un ensemble de sites dont les états ont une influence les uns sur les autres.

Définition 15 (Potentiel de Gibbs et Distribution de Gibbs) *Un potentiel de Gibbs sur Γ^S relatif à la topologie N est une famille $(V_C)_{C \subset S}$ de fonctions de Γ^S dans $\mathbb{R}$ tels que :*

1. *$V_C = 0$ si C n'est pas une clique.*
2. *Pour tout x et x' dans Γ^S et pour tout C dans S*

$$(x(C) = x'(C)) \Rightarrow (V_C(x) = V_C(x')) \tag{2.3}$$

La fonction d'energie ε dérive du potentiel V_C si :

$$\varepsilon(X) = \sum_C V_C(x). \tag{2.4}$$

Dans ce contexte un champ ayant une distribution de la forme :

$$\pi_T(x) = \frac{1}{Z_T} e^{-\frac{1}{T}\varepsilon(x)} \tag{2.5}$$

est appelé ***champ de Gibbs***

Notons que cette distribution est courante chez les physiciens et que l'on peut démontrer que certains systèmes physiques suivent cette loi.

Enfin voici des résultats mathématiques nous permettant de travailler sur les deux notions (Gibbs et Markov) et que nous utiliserons ensuite.

2.5 Résultats

Les deux résultats importants sont qu'un champ de Gibbs est un unique champ de Markov et que, si on fixe un minimum du potentiel, un champ de Markov est aussi un unique champ de Gibbs.

2.5.1 Du global au local

Théorème 1 *Si X est un champ de Gibbs relativement à une topologie alors c'est un champ de Markov relativement à la même topologie et de plus la spécification locale nous est donnée par :*

$$\pi^S(x) = \frac{e^{-\sum_{C \ni s} V_C(x)}}{\sum_{\gamma \in \Gamma} e^{-\sum_{C \ni s} V_C(\gamma, x(S \setminus s))}} \tag{2.6}$$

2.5.2 Du local au global

Théorème 2 (d'Hammersley-Clifford) *Un champ de Markov (satisfaisant une condition dite de "positivité") est un champ de Gibbs et a donc une distribution de la forme (2.5) avec une énergie dérivant d'un potentiel.*

L'expression exacte du potentiel est connue mais est assez compliquée (elle sera détaillée et utilisée en Chapitre 3).

2.5.3 Idées pour la démonstration

La preuve de la partie Gibbs implique Markov est simple, et constructive.

Par contre celle du théorème de Hammersley et Clifford est plus compliquée, elle utilise notamment la formule d'inversion de Moebius. Elle est due à Grimmett en 1973, qui en donne une preuve assez constructive une fois qu'on a la formule d'inversion de Moebius (On construit les potentiels).

2.6 Exemples d'utilisation

2.6.1 Chaines de Markov

Aprés avoir rapidement vérifié qu'une chaîne de Markov est un champ de Markov, il est donc intéressant grâce au théorème 2 de trouver une énergie et une distribution relative à cette chaîne. Les calculs ne sont pas très longs et des exemples simples peuvent donner une énergie rapidement.

2.6.2 Modèle d'Ising

Le modèle d'Ising est celui qu'on a défini en début de chapitre. Il est clair qu'une fois qu'on fixe la topologie (Voisinages, Cliques...), ce modèle représente un champ de Gibbs. Le théorème s'applique donc et on en déduit une loi de spécification locale, dont la formule assez compliquée s'obtient directement avec le théorème.

2.6.3 Traitement d'images

L'utilisation du théorème ici est moins directe et nécessite donc un peu plus d'explications.

Enjeux

Le but est, étant donnée une image avec des trous, ou brouillée, de réussir à la restaurer complétement en utilisant les parties claires.

On suppose donc, étant donnée une topologie, que cette image est un champ de Markov. Ce postulat est fondé sur le fait que la couleur de chaque pixel dépend fortement de celles de ses proches voisins. Il est donc d'usage de definir le voisinage d'un point de l'image par l'ensemble des points étant à une distance inférieure à une constante. Le problème est que la spécification locale ne nous est pas donnée et qu'il est nécessaire de l'obtenir pour compléter l'image.

Méthode

Le fait que le théorème de Hammersley Clifford existe nous donne l'idée de chercher une distribution globale de l'image sous la forme d'un champ de Gibbs. On détermine donc une formule heuristiquement pour l'énergie et on la fait dépendre de paramètres. Ensuite par des méthodes statistiques (maximum de vraisemblance...) on essaye de les ajuster au mieux pour que ca coincide avec la ou les images observées.

Enfin, l'autre sens du théorème nous permet de retrouver la spécification locale, qui est donnée directement en fonction de l'énergie

Chapitre 3

Application aux algorithmes distribués

Cette partie contient l'essentiel du travail de ce stage : Nous tenterons ici de donner une méthode générale de démonstration de l'auto-stabilisation d'algorithmes (avec des conditions bien précises) en utilisant les résultats du chapitre précédent.

3.1 Introduction

L'idée principale est de changer d'objet pour travailler sur les algorithmes distribués. En effet, la ressemblance entre les lois qui régissent un réseau de spins et celles qui donnent l'évolution d'un anneau de machines dans un algorithme distribué nous ont fait nous intéresser à l'ensemble des états pris par les machines a posteriori après un certain nombre (fixé) d'étapes de l'algorithme.

Plus concrètement, on écrit une matrice de taille (m, N) où m est le nombre d'étapes et N le nombre de machines, en écrivant ligne par ligne l'évolution du système sous l'effet de l'algorithme considéré. On obtient donc une matrice de la forme suivante :

$$\mathcal{M}_m = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

Dans laquelle le terme de la l^{eme} ligne et de la i^{eme} colonne que l'on notera l, i représente l'état de la machine i après l étapes de l'algorithme.

Pour commencer, il est nécessaire de donner quelques définitions permettant de raisonner avec ces matrices.

Définition 16 (Matrices $\mathcal{L}$-satisfaisantes) *Une matrice (m, N) est dite satisfaisante si ses deux dernières lignes sont des lignes appartenant à la classe $\mathcal{L}$.*

Dans l'exemple de Herman, la classe L est la classe des lignes n'ayant qu'un seul jeton.

Nous les appellerons les matrices $\mathcal{L}^{ok}$ et nous noterons leur ensemble $\mathcal{U}_m^{ok}(T_0)$ où T_0 est l'état initial de l'algorithme.

De telles matrices sont donc de la forme suivante :

$$\mathcal{M}_m^{ok} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

Définition 17 (Matrices non $\mathcal{L}$-satisfaisantes) *Une matrice (m, N) est dite non$\mathcal{L}$-satisfaisante lorsqu'elle n'est pas satisfaisante, c'est à dire lorsque ses deux dernières ne sont pas de la bonne forme. Elle sera noté $\mathcal{M}_m^{nok}$ et leur ensemble $\mathcal{U}_m^{nok}(T_O)$.*

Ensuite, il s'agit de considérer l'ensemble des matrices (à m fixé) et de définir une distribution dessus, en établissant que la probabilité de chaque matrice est la probabilité que les sites aient suivi l'évolution décrite par la matrice.

3.2 Méthodologie générale et résultats

Nous allons donc travailler sur ces nouveaux objets et de démontrer les propriétes d'auto-stabilisation sur eux-mêmes. Nous considérons dans le cas général un algorithme distribué synchrone et dans lequel chaque machine évolue en fonction de son état et de l'état de son voisin de gauche (on peut généraliser , bien sur, au cas où c'est son voisin de droite).
Les propriétés et théorèmes seront énoncés dans le cas général, mais certaines démonstrations nécessitant un calcul précis seront faites en prenant l'exemple de l'algorithme de Herman.

Pour cela nous allons utiliser le théorème de Hammersley et Clifford et suivre la méthodologie suivante :

3.2.1 Les matrices sont des champs de Markov

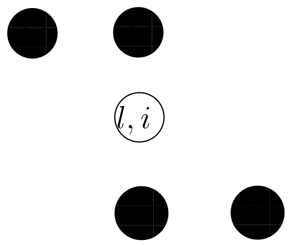
On considère que m est fixé. On construit comme décrit ci-dessus les matrices représentant l'évolution du système sous l'effet de l'algorithme. La première remarque que l'on peut formuler est la suivante :

Propriété 1 *L'ensemble des matrices de taille (m,N) générées par un algorithme synchrone est un champ de Markov.*

La preuve est assez évidente, chaque machine ne va évoluer qu'en fonction de son état et de celui de son voisin (de gauche dans l'exemple de Herman) ; les valeurs de la matrice ne dépendent donc que d'un petit nombre de voisins dont voici un détail plus formel :

Preuve : Dans notre modèle on a :
- L'ensemble S des sites est l'ensemble des éléments de la matrice
- L'espace Γ des phases est celui des valeurs que peuvent prendre un machine de l'algorithme (ici 0 ou 1).
- Le champ aléatoire est donné par l'ensemble des variables aléatoires $\mathcal{M}_m(l,i)$ à valeurs dans Γ.

L'algorithme nous suggére le voisinage à prendre, en effet, chaque site ne dépend que de l'état de son passé t du passé de son voisin de gauche, soit par nécessité de symétrie le voisinage suivant :



Les cliques sont donc le singleton $\{l,i\}$ et les couples $\{(l,i);(l-1,i)\}$ et $\{(l,i);(l-1,i-1)\}$.
Ainsi le système étudié est bien un champ de Markov.

La suite consiste à essayer de trouver la distribution globale de cet ensemble de matrices, en passant de la notion de Champs de Markov à celle de Champs de Gibbs, via le théorème de Hammersley et Clifford et de travailler sur les énergies obtenues.

3.2.2 Les matrices sont des champs de Gibbs

La partie suivante décrira un algorithme modifié nous permettant de passer au Champ de Gibbs.

L'algorithme de Herman modifié : l'ε-algorithme

Pour pouvoir appliquer le théorème de Hammersley et Clifford il faut que la condition de positivité soit vérifiée. Les transitions déterministes de l'algorithme original rendent cette condition fausse et donc l'équivalence Markov-Gibbs inapplicable, nous devons donc le modifier en remplaçant toutes les transitions impossibles (probabilité nulle) en des transitions de probabilité très faible et qui dépendent de l'étape l de l'algorithme (ε_l), cela suffira à impliquer la condition de positivité.

Par exemple dans le cas de Herman voici l'algorithme que nous utiliserons :

- Si $x_i = x_{i-1}$ alors $x_i' = \gamma_i$
- Si $x_i \neq x_{i-1}$ alors $x_i' = (1 - \epsilon_l) * x_{i-1} + \epsilon_l * x_i$

où ϵ_l est la variable aléatoire qui vaut 1 avec une probabilité ε_l et 0 sinon. ϵ_l varie en fonction de l ; nous verrons quelle forme lui donner plus tard, mais pour l'instant il suffit de considérer que la suite ϵ_l décroit (nous verrons ensuite qu'il faut qu'elle décroisse assez fortement).

Calcul des potentiels et de l'énergie

Nous pouvons donc à présent appliquer le théorème, les matrices découlant de l'algorithme modifié sont bien des champs de Gibbs, et donc la propriéte suivante découle directement de la forme (Gibbs) que va prendre la distribution des matrices.

A partir d'ici, nous travaillerons essentiellement sur les énergies de matrices pour démontrer les propriétés de l'algorithme. Pour cela nous avons besoin de la propriété suivante qui découle directement de la forme des distributions de Gibbs.

Propriété 2 *La somme des potentiels locaux (Energie) est minimale si et seulement si la matrice est la plus probable.*

Nous devons donc maintenant savoir quand cette énergie est minimale, pour cela il nous faut calculer (dans notre cas particulier) les potentiels sur les cliques en partant des probabilités conditionnelles.

Les cliques ont la même forme pour tous les algorithmes ayant les propriétés voulues ; les probabilités conditionnelles sont faciles à calculer ; elles correspondent directement des commandes de l'algorithme sur les machines. Nous donnons ici, le calcul exact dans le cas de Herman.

Loi de spécification locale

Notations

Nous noterons :

$$\Pi_{l,i}\begin{pmatrix} \alpha & \beta \\ . & \gamma \end{pmatrix}$$

la probabilité condtionnelle que le machine i soit dans l'état γ à l'étape l sachant que $(i-1)$ vaut α et que i vaut β à l'étape $l-1$

Les probabilités nous sont données directement par l'algorithme par les commandes :

Si $x_{i-1} = \alpha$ et $x_i = \beta$ alors $x'_i = \gamma$, soit :

Exemple : L'ε-algorithme de Herman

$\Pi_{l,i}\begin{pmatrix} 0 & 0 \\ . & 0 \end{pmatrix} = 1/2$

$\Pi_{l,i}\begin{pmatrix} 0 & 0 \\ . & 1 \end{pmatrix} = 1/2$

$\Pi_{l,i}\begin{pmatrix} 0 & 1 \\ . & 0 \end{pmatrix} = 1 - \varepsilon_l$

$\Pi_{l,i}\begin{pmatrix} 0 & 1 \\ . & 1 \end{pmatrix} = \varepsilon_l$

$\Pi_{l,i}\begin{pmatrix} 1 & 0 \\ . & 0 \end{pmatrix} = \varepsilon_l$

$\Pi_{l,i}\begin{pmatrix} 1 & 0 \\ . & 1 \end{pmatrix} = 1 - \varepsilon_l$

$\Pi_{l,i}\begin{pmatrix} 1 & 1 \\ . & 0 \end{pmatrix} = 1/2$

$\Pi_{l,i}\begin{pmatrix} 1 & 1 \\ . & 1 \end{pmatrix} = 1/2$

Valeur des potentiels

La formule qui donne les potentiels sur les cliques en fonction de la spécification locale est :

$$V_A(\mathcal{M}) = -\sum_{B\subset A}(-1)^{|A-B|}ln(\Pi^s(\mathcal{M}^B)).$$

Ici A est une clique et la somme se fait sur tous les ensembles inclus dans la suite. s est un élément de A fixé à l'avance (dans notre cas se sera (l,i)). Ce qui nous donne à calculer pour chaque (l,i) les trois potentiels suivants :

$$V_{l,i}(\mathcal{M}) = -ln\Pi_{l,i}\begin{pmatrix} 0 & 0 \\ . & \mathcal{M}_{l,i} \end{pmatrix}$$

$$V_{(l-1,i),(l,i)}(\mathcal{M}) = ln\Pi_{l,i}\begin{pmatrix} 0 & 0 \\ . & \mathcal{M}_{l,i} \end{pmatrix} + ln\Pi_{l,i}\begin{pmatrix} 0 & \mathcal{M}_{l-1,i} \\ . & 0 \end{pmatrix} - ln\Pi_{l,i}\begin{pmatrix} 0 & \mathcal{M}_{l-1,i} \\ . & \mathcal{M}_{l,i} \end{pmatrix}$$

$$V_{(l-1,i-1),(l,i)}(\mathcal{M}) = ln\Pi_{l,i}\begin{pmatrix} 0 & 0 \\ . & \mathcal{M}_{l,i} \end{pmatrix} + ln\Pi_{l,i}\begin{pmatrix} \mathcal{M}_{l-1,i-1} & 0 \\ . & 0 \end{pmatrix} - ln\Pi_{l,i}\begin{pmatrix} \mathcal{M}^{l-1,i-1} & 0 \\ . & \mathcal{M}_{l,i} \end{pmatrix}$$

Ensuite, pour simplifier le calcul des énergies, on rassemble les trois potentiels s'intéressant à i^l et on note W la valeur de cette somme sur les motifs correspondants.

Soit

$$W = V_{l,i} + V_{(l-1,i-1),(l,i)} + V_{(l-1,i),(l,i)}$$

L'énergie finale sera donc calculée en sommant ces termes sur tous les éléments de la matrice.

Soit plus précisement dans le cas de l'algorithme de Herman modifié :

Exemple : L'ε-algorithme de Herman

$W\begin{pmatrix} 0 & 0 \\ . & 0 \end{pmatrix} = -ln2$

$W\begin{pmatrix} 0 & 0 \\ . & 1 \end{pmatrix} = -ln2$

$W\begin{pmatrix} 0 & 1 \\ . & 0 \end{pmatrix} = -ln2$

$W\begin{pmatrix} 0 & 1 \\ . & 1 \end{pmatrix} = ln(\frac{(1-\varepsilon_l)}{\varepsilon_l}) - ln2$

$W\begin{pmatrix} 1 & 0 \\ . & 0 \end{pmatrix} = -ln2$

$W\begin{pmatrix} 1 & 0 \\ . & 1 \end{pmatrix} = ln(\frac{\varepsilon_l}{(1-\varepsilon_l)}) - ln2$

$W\begin{pmatrix} 1 & 1 \\ . & 0 \end{pmatrix} = -ln2$

$W\begin{pmatrix} 1 & 1 \\ . & 1 \end{pmatrix} = -ln2$

Notation :

Nous noterons
$$\lambda_l = \frac{\varepsilon_l}{(1-\varepsilon_l)}$$

Calcul des énergies optimales (dans l'exemple de Herman)

Calcul de l'énergie de $\mathcal{M}_m^{ok}$ (pour la plus énergétique des $\mathcal{M}_m^{ok}$) :

Remarquons tout d'abord qu'il y a autant de potentiels locaux à sommer que de termes dans la matrices et qu'il n'existe que trois valeurs pour ces potentiels : $-ln(2)$, $ln(\lambda_l) - ln(2)$ et $-ln(\lambda_l) - ln(2)$. Le fait que ε_l soit compris entre 0 et $\frac{1}{2}$ nous donne :

$$-ln(\lambda_l) - ln(2) > 0 > -ln(2) > ln(\lambda_l) - ln(2).$$

A la m^{me} ligne, la forme de la matrice est connue et donc nous sommons $\frac{N-1}{2}$ fois $ln(\lambda_m)$ et $\frac{N+1}{2}$ fois $-ln(2)$. Avant cette ligne il nous suffit de considérer que l'on est dans le pire des cas (Energie la plus forte) et que l'on ait $\frac{N-1}{2}$ fois $-ln(\lambda_i)$ par ligne, ce qui nous donne en définitive :

$$\mathcal{E}(\mathcal{M}_m^{ok}) \leq (\frac{N-1}{2})ln(\lambda_m) - (\frac{N}{2})mln(2) - \frac{N-1}{2}\sum_{l=2}^{m-1} ln(\lambda_l).$$

On a l'égalité pour la matrice suivante :

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & 1 & 0 & \cdots \\ & & \cdots & & & \\ 0 & 1 & 0 & 1 & 0 & \cdots \\ 1 & 0 & 1 & 0 & 1 & \cdots \end{pmatrix}$$

On notera cette énergie maximale $Max(\mathcal{E}_m^{ok})$

Calcul de l'énergie de $\mathcal{M}_m^{nok}$ (pour la moins énergétique des $\mathcal{M}_m^{nok}$) :

Pour cela, il nous faut le maximum de potentiels locaux valant $ln(\lambda_l)$ et le minimum valant $-ln(\lambda_l)$, soit le maximum de $\begin{pmatrix} 1 & 0 \\ . & 1 \end{pmatrix}$ et le minimum de $\begin{pmatrix} 0 & 1 \\ . & 1 \end{pmatrix}$.

Par ligne il n'est possible d'avoir que $\frac{N-1}{2}$ $\begin{pmatrix} 1 & 0 \\ . & 1 \end{pmatrix}$.

Ainsi dans les $m-1$ premières lignes, on pourra voir uniquement des motifs de potentiel minimum et sur la dernière ligne il y en aura un de moins

(en effet la matrice doit être non satisfaisante)

Dans ce cas, nous pouvons calculer l'énergie de la matrice suivante :

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 1 \\ & & & \cdots & & & & \\ 0 & 1 & 0 & \cdots & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & \cdots & 0 & 1 & 1 & 1 \end{pmatrix}$$

Ce qui nous donne ce résultat :

$$\mathcal{E}(\mathcal{M}_m^{nok}) \geq (\frac{N-3}{2})ln(\lambda_m) - (mN+1)ln(2) + \frac{N-1}{2}\sum_{l=2}^{m-1} ln(\lambda_l).$$

avec égalité pour la matrice ci-dessus. Nous noterons ce minimum d'énergie $Min(\mathcal{E}_m^{nok})$.
Il nous reste maintenant à exploiter ces valeurs et à essayer d'en déduire des propriétés sur les matrices.

3.2.3 Convergence et Auto-stabilisation

Théorème 3 (Condition de convergence) *Si on a :*

$$\lim_{m\to+\infty} Min(\mathcal{E}_m^{nok}) - Max(\mathcal{E}_m^{ok}) - mNln(2) = +\infty$$

alors l'ε-algorithme est convergent soit :

$$\lim_{m\to+\infty} \sum_{\mathcal{U}_m^{ok}(T_0)} P(\mathcal{M}_m^{ok}) = 1$$

Nous allons plutôt démontrer l'égalité suivante qui est évidemment équivalente à la condition donnée dans le théorème :

$$\lim_{m\to+\infty} \sum_{\mathcal{U}_m^{nok}(T_0)} P(\mathcal{M}_m^{nok}) = 0$$

Preuve

Le fait que $Min(\mathcal{E}_m^{nok}) - Max(\mathcal{E}_m^{ok})$ tende vers $+\infty$ établit que la probabilité de finir par des éléments de $\mathcal{L}$ et bien plus grande que celle de finir avec plus d'un jeton. Mais ce n'est pas le résultat escompté. Nous devons sommer sur l'ensemble des matrices non satisfaisantes et vérifier que leur probabilité tend bien vers 0.

On peut remarquer qu'il y a, à m fixé, $2N2^{(m-2)N}$ matrices satisfaisantes (liberté de choix pour les premières lignes) et donc on peut majorer le nombre de matrices non satisafaisantes par 2^{mN}.

De ce fait, pour simplifier les calculs, nous allons utiliser cette majoration et minorer l'énergie de chaque matrice par le minimum calculé précedemment.

Ainsi

$$\sum_{\mathcal{U}_m^{nok}(T0)} P(\mathcal{M}_m^{nok}) < 2^{mN} e^{-Min(\mathcal{E}_m^{nok})}$$

soit on a à démontrer que
$\frac{\sum_{\mathcal{U}_m^{nok}} P(\mathcal{M}_m^{nok})}{P(\mathcal{M}_m^{ok})}$ tend vers 0, donc en passant aux logarithmes que :

$$\lim_{m\to+\infty} Min(\mathcal{E}_m^{nok}) - Max(\mathcal{E}_m^{ok}) - mNln(2) = +\infty$$

Ainsi si cet égalité est vérifiée le rapport de probabilités tend vers 0 et donc le numérateur tend vers 0 ainsi :

$$\lim_{m\to+\infty} \sum_{\mathcal{U}_m^{nok}(T0)} P(\mathcal{M}_m^{nok}) = 0$$

Théorème 4 *Pour un choix correct de ε, l'ε-algorithme de Herman est convergent.*

Preuve(dans l'exemple de Herman) :

Rappelons les valeurs de $Min(\mathcal{E}_m^{nok})$ et de $Max(\mathcal{E}_m^{ok})$:

$$Max(\mathcal{E}(m^{ok}) = (\frac{N-1}{2})ln(\lambda_m) - Nmln(2) - \frac{N-1}{2}\sum_{l=2}^{m-1} ln(\lambda_l)$$

.

$$Min(\mathcal{E}_m^{nok}) = (\frac{N-3}{2})ln(\lambda_m) - (mN+1)ln(2) + \frac{N-1}{2}\sum_{l=2}^{m-1} ln(\lambda_l)$$

.

$$Min(\mathcal{E}_m^{nok}) - Max(\mathcal{E}_m^{ok}) - mNln(2) = -ln\lambda_m + (N-1) + \sum_{l=2}^{m-1} ln\lambda_l - (mN+1)ln2$$

Dans ce cas il suffit de choisir λ_m par récurrence de telle sorte que cette somme tende vers $+\infty$ avec m. Il faut donc que λ_m décroisse très rapidement (par exemple $\frac{1}{2^{2^{2m}}}$). On obtient ainsi la condition suffisante de convergence du théorème précédent.

L'ε-algorithme de Herman est donc convergent.

Théorème 5 (d'auto-stabilisation) *Si pour une certaine suite ε l'ε-algorithme est convergent alors l'algorithme original est convergent.*

Preuve(dans l'exemple de Herman) :

Ici nous allons comparer les probabilités de 2 matrices, dont l'une représente l'avancement du système pour l'algorithme original et l'autre représente l'avancement du système pour l'ε-algorithme.
Nous noterons donc Π_0 la loi de probabilité sur les matrices correspondants à l'algorithme original et Π_ε la probabilité pour celles de l'ε-algorithme.

Soit pout toute matrice $\mathcal{M}$:

$$\Pi_0(\mathcal{M}_m^{nok}) \leq \frac{\Pi_\varepsilon(\mathcal{M}_m^{nok})}{(1-\varepsilon_1)^{\frac{N+1}{2}}...(1-\varepsilon_m)^{\frac{N+1}{2}}}$$

Ce résultat est du au fait que la différence entre les deux algorithmes se fait lors des transitions déterministes de l'algorithme original, et cela ne peut arriver que $\frac{N+1}{2}$ fois par ligne.

On doit choisir la suite ε qui décroit assez vite de telle sorte que le produit $(1-\varepsilon_1)^{\frac{N+1}{2}}...(1-\varepsilon_m)^{\frac{N+1}{2}}$ tende en décroissant vers une limite finie L non nulle, et donc on obtient en sommant :

$$\sum_{\mathcal{U}_m^{nok}(T_0)} P^0(\mathcal{M}_m^{nok}) \leq \frac{1}{L} \sum_{\mathcal{U}_m^{nok}(T_0)} P^\varepsilon(\mathcal{M}_m^{nok}).$$

Enfin, en faisant tendre m vers l'infini, on obtient la convergence du terme de gauche vers 0 et donc la convergence de l'algorithme original.

Corollaire 1 *Si l'ε-algorithme vérifie la condition de convergence (ie : $\lim_{m\to+\infty} Min(\mathcal{E}_m^{nok}) - Max(\mathcal{E}_m^{ok}) - mNln(2) = +\infty$), l'algorithme original est auto-stabilisant.*

En effet la propriété de fermeture est évidemment vérifiée pour l'algorithme original.

Conclusion

Nous avons donc, durant ce stage, proposé une nouvelle méthode , basée sur les principes de Physique Statistique, pour prouver l' auto-stabilisation d'algorithmes distribués. Nous utilisons des matrices représentant l'historique de l'état des machines au lieu de travailler uniquement sur les configurations, et nous introduisons une notion d'énergie sur ces matrices correspondant à une énergie de Gibbs.

Cette méthode a été prouvée en partie sur l'exemple de Herman, mais d'autres (comme celui de Flatebo et Datta) peuvent être prouvés de la même manière.

Une autre manière d'étendre nos recherches serait de regarder d'autres topologies de réseaux de machines, ou de s'intéresser à des algorithmes où chaque machine a différents voisinages (gauche ET droite par exemple). L'étude de ces cas ne devrait pas poser de problèmes théoriques mais des problèmes liés aux calculs (notamment des énergies optimales).

Bibliographie

[Br9] P. Brémaud. *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues.* Springer, 1999.

[DFP01] M. Duflot, L. Fribourg, and C. Picaronny. Randomized distributed algorithms as Markov chains. In Springer, editor, *Proc. 15th Int. Conf. on Distributed Computing (DISC'2001), LNCS 2180*, pages 240–254, 2001.

[Dij74] E.W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11) :643–644, Nov. 1974.

[Dol00] S. Dolev. *Self-Stabilization.* The MIT Press, 2000.

[FD94] M. Flatebo and A.K. Datta. Two-state self-stabilizing algorithms for token rings. *IEEE Transactions on Software Engineering*, 20(6) :500–504, June 1994.

[Her90] T. Herman. Probabilistic self-stabilization. *IPL*, 35(2) :63–67, June 1990.

[LR81] D. Lehmann and M. O. Rabin. On the advantages of free choice : a symmetric and fully-distributed solution to the dining philosophers problem. In *Proc. 8th Annual ACM Symp. on Principles of Programming Languages (POPL'81)*, pages 133–138, 1981.

[Lyn97] N.A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, Inc., 1997.

[PF91] B. Prum and J.-C. Fort. *Stochastic Processes on a Lattice and Gibbs Measures.* Kluwer Academic Publishers, 1991.