Méthodes probabilistes pour la vérification des systèmes distribués

Stéphane Messika

ENSC - 2004 N^o22

THÈSE

présentée à l'École Normale Supérieure de Cachan

pour obtenir le grade de

Docteur de l'École Normale Supérieure de Cachan

par : Stéphane MESSIKA

Spécialité : INFORMATIQUE

Méthodes Probabilistes pour la Vérification des Systèmes Distribués

Soutenue le 14 décembre 2004 devant un jury composé de :

– Joffroy BEAUQUIER	Rapporteur
– Laurent FRIBOURG	co-Directeur de thèse
– Marta KWIATKOWSKA	Examinatrice
– Richard LASSAIGNE	Rapporteur
– Satya MAJUMDAR	Examinateur
– Jean-François MONIN	Président du jury
– Claudine PICARONNY	co-Directrice de thèse

Remerciements

Merci Dina,
Tu m'as apporté pendant ces longues années d'études le soutien, le bonheur et l'amour qui m'ont donné le courage de mener à bien ce projet.

Merci Claudine et Laurent,
Vous m'avez donné les clefs et les moyens de réussir à rédiger cette thèse. Nos discussions souvent animées, les murs du LSV s'en souviendront, ont été grandement nécessaires pour me remettre sur de bons rails.

Merci Richard et Joffroy
En m'ayant fait l'honneur d'être rapporteurs de cette thèse, vous m'avez par vos nombreuses remarques, permis d'améliorer grandement ce manuscrit et les résultats qu'il présente.

Merci Jean-François, Satya et Marta
Votre présence dans mon jury de thèse est un honneur. Nos nombreuses discussions (même tardives) ont toujours attisé ma curiosité scientifique.

Merci Manuel, Muriel, Houda et Benjamin
A vous, je dois un grand merci, car vous m'avez supporté (parfois pendant toute la thèse, hein Manu) dans mes bons comme dans mes mauvais jours. Vous avez été toujours là pour me soutenir, ou pour répondre à toutes mes questions nulles sur UNIX, Latex ou autres.

Merci aux membres du LSV
Pour leur qualité d'attention exceptionnelle. J'ai souvent eu à discuter avec plusieurs d'entre vous, et vous avez toujours été présents. Ce sont ces qualités qui font de ce lieu un cadre idéal pour la recherche. En particulier, merci à Marie pour ses nombreux conseils et son aide précieuse

Merci à mes amis
Qu'ils m'excusent, si je ne les cite pas tous, mais les bons moments passés avec chacun d'entre eux durant cette thèse me reviennent souvent en tête.

Enfin, et surtout merci à ma famille
Tout d'abord à mes parents pour le soutien inconditionnel qu'ils m'ont donné et pour leurs conseils souvent douloureux mais toujours justes. Puis à mes deux frères (Gabriel et Benjamin) qui ont le grand mérite (encore plus que mes collègues de bureau) d'avoir supporté mes nombreux défauts pendant de longues années.

Et toi, Elie, je ne t'ai pas oublié, je te gardais pour la fin, merci de l'immense bonheur que tu me donnes depuis déjà plus de vingt mois.

Table des matières

Chapitre 1

Introduction

La théorie, c'est quand on sait tout et que rien ne fonctionne. La pratique, c'est quand tout fonctionne et que personne ne sait pourquoi. Ici, nous avons réuni théorie et pratique : rien ne fonctionne et personne ne sait pourquoi.
Albert Einstein

1.1 Le besoin de méthodes formelles de vérification

"Désolé, on a été coupé".

Chacun d'entre nous a déjà prononcé ou entendu de nombreuses fois cette phrase au cours d'une conversation téléphonique.
La croissance de l'utilisation des systèmes informatiques dans l'ensemble des domaines de la vie courante (transport, téléphonie, hi-fi, ...) pousse la communauté scientifique à s'intéresser de façon formelle à la question de leur fiabilité.
Assurer le bon fonctionnement des systèmes informatiques est devenu en effet une question cruciale dans de nombreux domaines, de par le danger induit (aéronautique, espace, systèmes de communications), ou le coût important de pannes (systèmes sur puces, automobile).
Au delà d'une garantie *qualitative* (sur le bon fonctionnement) il est souvent nécessaire de se poser des questions de performance (temps avant qu'une communication aboutisse, nombre moyens de pannes...), ce qui nous conduit également à étudier ces systèmes d'un point de vue *quantitatif*.
Pour ce faire, nous avons choisi dans une première partie de cette thèse d'utiliser des méthodes formelles de "model-checking" qui regroupent essentiellement trois phases : modélisation, spécification de la propriété à vérifier, vérification. Etant donné un système informatique, une première tâche est en effet de le modéliser. Il faut donc concevoir un modèle en utilisant des représentations classiques de l'informatique (notamment les différentes sortes d'automates) permettant de décrire fidèlement (selon un certain degré d'abstraction) les comportements du système qui nous intéresse. Ensuite, il faut traduire les propriétés requises du système (par exemple, le fait que les communications en cours avec un téléphone mobile ne soient jamais interrompues), en termes de formule logique. Enfin, il faut vérifier, en utilisant les techniques de model-checking, proprement dites, la validité de la formule dans le modèle élaboré.
Dans une deuxième partie de cette thèse, nous nous sommes intéressés à des méthodes de vé-

rification de propriétés de convergence, comme celles employées en physique statistique pour montrer le retour d'un système dynamique à un état d'équilibre après perturbation.

1.2 Les systèmes distribués probabilistes

1.2.1 Les systèmes distribués

La notion de système *distribué*, ou *réparti* englobe tous les systèmes composés de plusieurs entités (processus, ordinateurs, etc.) autonomes communiquant chacune avec ses "voisins" au moyen soit de canaux de communication soit de variables partagées. Chaque entité évolue alors en fonction de son état propre et de celui de ses voisins. Les réseaux comme Internet sont des exemples typiques de systèmes distribués.

L'enjeu des algorithmes distribués est de réussir à faire collaborer toutes ces entités pour accomplir une tâche globale. Pour ne donner qu'un exemple, on peut citer le consensus, qui consiste à mettre d'accord tous les composants du système, ou du moins ceux qui ne sont pas en panne, sur une certaine valeur.

Du fait que les entités ne communiquent qu'avec un voisinage restreint, aucun des participants à l'algorithme n'a une connaissance globale de l'état du système (appelé *configuration*), ce qui accroît la difficulté de l'algorithme et de sa vérification. De plus, on a rarement des informations sur l'ordre dans lequel les différentes entités vont effectuer des actions.

Classiquement, on modélise le fait que ce choix de l'ordre des processus n'est pas toujours prédéfini, en introduisant la notion de *politique*, qui représente un mécanisme abstrait externe, choisissant à chaque étape le ou les entités qui vont effectuer une action à un instant donné. La politique est généralement obligée de choisir des processus pouvant effectivement faire une action, mais parmi eux son choix peut être arbitraire. La politique symbolise donc la part de non-déterminisme présente dans certains systèmes distribués.

Les ouvrages [Tel91, Tel94, Lyn96] donnent une bonne vue d'ensemble sur l'étude des algorithmes distribués.

1.2.2 L'introduction des probabilités

"Dieu ne joue pas au dés." Cette réplique connue d'Einstein nous force à poser la question : pourquoi ajouter aux systèmes une complication supplémentaire par l'introduction de probabilités ? Il existe plusieurs raisons :

Modéliser l'environnement

Une des raisons les plus naturelles quant à l'introduction des probabilités est la non-maîtrise de l'environnement du système. On peut rarement décrire de manière exacte le comportement de l'environnement dans lequel le système est plongé.
On utilise donc des lois de probabilités établies souvent à l'aide de statistiques d'observation et qui permettent de décrire les comportements prévisibles de l'environnement. C'est alors la modélisation probabiliste de l'environnement (non probabiliste à la base) du système qui impose à la vérification de se faire par des méthodes probabilistes.

Exemple 1.1. *on peut considérer le cas des files d'attente. Le taux d'entrée dans la file est un facteur non maîtrisé. On choisit donc, souvent de le modéliser suivant une loi de Poisson. L'algorithme de gestion de la file, souvent non probabiliste, doit donc être analysé de manière probabiliste.* ♣

Résoudre des problèmes d'algorithmique distribuée

Un aspect des systèmes distribués homogènes est la symétrie de l'ensemble : tous les agents ont souvent la même tâche à accomplir ou le même algorithme à suivre. Dans ce cadre, obtenir de ce système un résultat dissymétrique (comme l'élection d'un leader par exemple) est souvent un problème insoluble. Lehmann et Rabin dans [RL94] prouvent ainsi que le problème connu sous le nom du problème des philosophes n'a pas de solution déterministe. L'introduction de probabilités dans de tels algorithmes permet de casser de telles symétries bloquantes. De nombreux autres systèmes nécessitent l'introduction de "transitions probabilistes" pour éviter d'entrer dans un état bloquant.
Dans cette thèse nous donnerons plusieurs exemples d'algorithmes pour lesquels l'absence d'états bloquants (ou la convergence vers un état récurrent) est assurée par l'introduction de comportements probabilistes (algorithmes d'évitement de collision dans les protocoles de communication ou de partage de ressources).

Exemple 1.2. *Le dilemme des philosophes est un problème d'allocation de ressources. N philosophes sont assis autour d'une table ronde, ils ont devant eux une assiette et deux baguettes (chacune partagée avec l'un de leurs voisins). Pour qu'ils puissent manger, ils ont besoin des deux baguettes à la fois. Lehmann et Rabin ont montré qu'il n'existait pas d'algorithme déterministe (distribué et uniforme) qui assure qu'un philosophe pourra manger au bout d'un temps fini. De plus ils ont conçu un algorithme probabiliste permettant d'obtenir ce résultat.* ♣

Réduire la complexité

La réduction de la complexité (en temps et en espace) semble être en contradiction avec l'introduction des probabilités, mais pourtant il est souvent moins coûteux d'utiliser un algorithme probabiliste qu'un algorithme déterministe (voir [MR95] et exemple 1.3).
Ainsi, il arrive souvent que des systèmes probabilistes soient plus performants que des systèmes déterministes, comme on peut le constater en étudiant les différents protocoles de communication. Beaucoup de protocoles font en effet appel à des tirages probabilistes sur les temps d'attente nécessaires avant l'envoi d'un message de façon à améliorer les propriétés du système (temps de réponse, temps de livraison d'un message avec succès, ...).

Exemple 1.3. *Le parcours de sommets sur un graphe est un problème classique en informatique tant ces applications sont nombreuses (recherche d'information, vérification du bon fonctionnement de toutes les machines). Un algorithme déterministe qui parcourt tous les sommets du graphe a besoin de mémoriser la taille N du graphe (il doit étiqueter les sommets déjà parcourus). Par contre une marche aléatoire sur les sommets (en passant de voisin en voisin) n'a besoin que d'un seul bit de mémoire. De plus la convergence de ces algorithmes est assez rapide, elle se fait en $O(N^3)$ dans le pire des cas (pour le graphe appelé lollypop).* ♣

1.2.3 Modèles

L'analyse des systèmes probabilistes a connu, ces dix dernières années, un essor considérable. Plusieurs types de méthodes de vérification ont été développés à l'aide d'outils classiques de théorie des probabilités.
Nous pouvons citer l'utilisation des chaînes de Markov et processus de décision markoviens pour la description des algorithmes auto-stabilisants probabilistes [BDLGJ02, Duf03, Dol00], ou le développement des algorithmes de model-checking des systèmes probabilistes et temporisés [dA97a, KNSS02].
Voici une classification des principaux types de modèles probabilistes utilisés préalablement à l'analyse des systèmes distribués. :

1. **Les chaînes de Markov :**

   Elles permettent de décrire des systèmes dans lesquels l'évolution ne dépend que de l'état courant du système mais pas du passé plus lointain (historique).
   Le modèle des chaînes de Markov est très utilisé dans la description des systèmes probabilistes. Parmi ses avantages, on peut citer sa simplicité de description ou les nombreux résultats connus sur les comportements (notamment asymptotiques) des chaînes.
   En outre, les systèmes se comportant comme des chaînes de Markov peuvent être vérifiés efficacement par des outils implémentant les algorithmes de model-checking (model-checkers) du type PRISM [PRI] ou APMC [APM] qui arrivent aujourd'hui à maturation.

2. **Les processus de décision markoviens :**

   Ils sont une extension des chaînes de Markov (voir [Put94]) qui permettent de combiner non déterminisme et probabilités ; plusieurs terminologies différentes correspondent à ce modèle, comme les *systèmes probabilistes et non déterministes* [BdA95] ou les *jeux à un joueur et demi* [CJH03].
   Ce modèle, classiquement utilisé en recherche opérationnelle ou en contrôle de processus industriels, est souvent bien adapté à la description synthétique des systèmes informatiques probabilistes. Le fait qu'on autorise à la fois le non-déterminisme et les probabilités permet, en effet, de modéliser de façon générique un grand nombre de politiques différentes associées à autant de chaînes de Markov.
   Les processus de décision markoviens peuvent aussi être vérifiés par le model-checker PRISM [PRI], mais la taille réduite des modèles acceptés limite a priori le nombre de cas d'études vérifiables en pratique.

3. **Les automates temporisés probabilistes :**

   Ils sont une extension des processus de décision markoviens et des automates temporisés. Il s'agit du modèle le plus général considéré ici : il permet en effet de prendre en considération l'évolution continue du temps dans le système étudié. On peut citer par exemple [KNSS02].
   L'automate temporisé probabiliste permet de décrire les systèmes où l'évolution du temps joue un rôle majeur comme dans les protocoles de communications.

La vérification de tels systèmes repose sur des techniques alliant les algorithmes du model-checking temporisé [AD94] (construction de graphe des régions, discrétisation,...) et les techniques de vérification probabilistes. Ils fonctionnent en transformant ces automates temporisés probabilistes en processus de décision markoviens qui sont vérifiés alors selon les méthodes classiques.

1.2.4 Propriétés

Dans cette thèse nous nous sommes intéressés essentiellement à la vérification de deux types de propriétés : les propriétés d'*accessibilité* et les propriétés de *convergence*.

1. Les propriétés d'accessibilité sont du type :

   "Etant donnée une configuration initiale du réseau, l'appel sera reçu en moins de 20 secondes avec probabilité 0.9"

   ou

   "Etant donnée une configuration initiale du réseau, le nombre moyen de collisions de messages avant la réception du message par le destinataire est supérieur à 7"

   Elles expriment le fait qu'un état "recherché " est toujours accessible, ou que des événements "satisfaisants" (envoi d'un appel) arriveront avant un temps donné.
   Les algorithmes de model-checking sur ce domaine sont récents [KNSS02] et les model-checkers ne permettent pas encore d'intégrer automatiquement tous les facteurs à la fois (temps, probabilité, non-déterminisme). La vérification de ces propriétés nécessite donc encore un travail de connaissance profonde du système, de modélisation et de combinaison manuelle de différentes techniques.

2. Les propriétés de convergence du type :

   "Quelle que soit la configuration initiale, le système atteint un ensemble de configurations satisfaisantes en un nombre fini d'étapes avec probabilité 1"

   Ces propriétés décrivent la robustesse d'un système. Elles assurent que, presque sûrement, le système se comportera de façon satisfaisante au bout d'un temps fini. Dans cette thèse, nous avons étudié non seulement des techniques de vérification de propriété de convergence mais aussi des méthodes de calcul du temps de convergence.
   Nous utilisons, notamment, plusieurs résultats mathématiques de la théorie des probabilités pour exhiber des techniques génériques, simples et efficaces pour prouver la convergence et estimer le temps de convergence.

1.3 Contributions de la thèse

Nos contributions s'articulent autour de deux grands axes principaux :

- La vérification de propriétés d'accessibilité sur les processus de décision markoviens.

- La vérification de propriétés de convergence sur les chaînes de Markov.

1.3.1 Premier axe : Accessibilité dans les systèmes temporisés probabilistes

Notre principale contribution dans la première partie consiste en l'analyse d'un protocole réel de télécommunication (CSMA/CD). Ce travail s'est essentiellement effectué dans le cadre du projet RNTL *AVERROES*. Une partie de ce travail s'est effectuée en étroite collaboration avec des équipes du LRI (Orsay) et de Birmingham [DFH+04, FMP04d].

Plus précisément, le travail a consisté à :
- Modéliser plusieurs variantes du protocole sous forme d'automates temporisés probabilistes, et d'en déduire, par discrétisation du temps notamment, des processus de décision markoviens équivalents.
- Analyser des probabilités extrémales ou des temps moyens pour différentes propriétés d'accessibilité, étant donnés des états de départ et d'arrivée appropriés : par exemple, temps moyen pour qu'un message entré en collision arrive finalement avec succès à destination.

Ces modèles ont été analysés avec les outils de model checking probabiliste PRISM [PRI] et APMC [APM], dont les avantages respectifs ont ainsi été comparés pour la première fois.

La présentation de cette étude de cas est précédée d'une explication théorique des principes de fonctionnement des outils de vérification employés, spécialement PRISM, et de la réduction du problème d'accessibilité dans les Processus de Décision Markoviens à un problème de recherche de plus court chemin dans un graphe stochastique.

1.3.2 Second axe : Convergence de systèmes paramétrés probabilistes

La contribution essentielle de la seconde partie consiste à utiliser de façon originale une technique probabiliste appelée "coupling", afin de démontrer la convergence (auto-stabilisation) de systèmes distribués probabilistes modélisés sous forme de chaînes de Markov. Cette contribution sur les liens entre coupling et auto-stabilisation a notamment fait l'objet d'une communication à DISC 2004 [FMP04b], qui a obtenu le "best student paper award".

Rappelons que dans une chaine de Markov, il n'y a pas non-déterminisme : en d'autres termes, il s'agit d'un Processus de Décision Markovien dont la politique est fixée une fois pour toutes. En revanche, dans cette partie, les systèmes seront considérés de façon générique, l'une de leurs dimensions (typiquement, le nombre N de machines mises en communication) étant paramétrée. La propriété étudiée de convergence signifie que, quel que soit l'état de départ, on est assuré d'atteindre un état satisfaisant (ou "légal") en un temps fini presque sûrement. On étudie également le temps moyen de convergence vers un état légal. La technique de coupling et un de ses raffinements ("path coupling") permettent de produire des preuves plus simples ou d'aboutir à de meilleurs bornes sur le temps de convergence.

De façon plus générale, nous considérons ensuite le problème de convergence (ou d'auto-stabilisation) d'un système distribué comme un problème d'évolution d'un système stochastique vers un état d'équilibre unique, ce qui le rapproche d'un problème de physique statistique, appelé problème de (absence de) transition de phase. Cela nous conduit à explorer d'autres techniques que le coupling, comme celle du "chemin de désaccord", utilisées couramment en physique Statistique [FMP03].

1.4 Plan de la thèse

Cette thèse est découpée en 4 parties :

1. Une introduction aux processus de décisions markoviens, dans laquelle on rappellera les résultats classiques de théorie des processus stochastiques (chapitre 2), puis nous décrirons le problème du plus court chemin stochastique ainsi que ses applications à la vérification de propriétés d'accessibilité (chapitre 3).

2. Une analyse des propriétés d'accessibilité des systèmes temporisés probabilistes (Axe 1). Cette partie comprend le chapitre 4 définissant formellement les automates temporisés probabilistes et donnant les différentes techniques de vérification. Le chapitre 5 décrit les résultats que nous avons obtenus dans le cadre de la vérification du protocole CSMA/CD suivant différentes méthodes et en utilisant différents outils. Cette partie retrace les articles [DFH+04, FMP04d].

3. Une étude de la convergence des systèmes paramétrés probabilistes (Axe 2). Le chapitre 6 donne les définitions préalables nécessaires à l'application du coupling et exprime les résultats obtenus dans [FMP04c] dans le cadre de l'application de ces techniques à un exemple issu de la physique statistique. Le chapitre 7 détaille, quant à lui, l'ensemble des critères permettant de traiter les problèmes d'auto-stabilisation à l'aide des techniques issues du coupling ; on peut retrouver l'ensemble de ces résultats dans [FMP04b]. Enfin, le chapitre 8 décrit les analogies faites entre les algorithmes distribués probabilistes vus comme des champs de Markov et les problèmes de physique statistique et évoquées dans [FMP03].

4. Un bilan général dans lequel nous donnerons quelques extensions et perspectives concrètes liées aux méthodes de preuve développées (chapitre 9) avant de conclure au chapitre 10.

Première partie

Introduction aux processus de décision markoviens

Chapitre 2

Introduction aux modèles stochastiques

Je ne joue plus à la roulette russe car je perds tout le temps.
Daniel Vadet

Les systèmes que nous allons étudier par la suite sont tous probabilistes, il est donc nécessaire de présenter les outils nécessaires à leurs études. Le premier (chaînes de Markov) est un modèle classique en théorie des probabilités et il existe une imposante littérature [Bré99, KSK66] traitant de ses différentes propriétés, nous ne donnerons ici que les résultats qui nous seront utiles par la suite. Pour le second modèle (processus de décision markoviens), il existe beaucoup moins d'oeuvres les décrivant (citons tout de même [Put94]). Nous essaierons donc ici, sans pour autant être exhaustifs, de détailler un peu plus les résultats importants concernant les processus de décision markoviens et d'établir des notions et des notations utiles pour la suite. Dans ce chapitre, nous adopterons le formalisme utilisé dans les thèses de Luca de Alfaro [dA97a] et de Marie Duflot [Duf03].

2.1 Une goutte de théorie des probabilités

Cette section a pour but de présenter, dans un cadre formel, une base théorique suffisante pour pouvoir définir les deux modèles stochastiques que l'on va utiliser dans les chapitres suivants : chaînes de Markov et processus de décision markoviens. Elle ne prétend pas donner une introduction exhaustive de la théorie des probabilités. Les références en la matière ne manquent pas. Parmi elles on peut citer l'ouvrage [Bré99] d'où vient la terminologie utilisée ici. On peut aussi regarder l'article [Pan01] où les probabilités sont présentées pour les informaticiens, et plus précisément pour le cadre des systèmes concurrents.

Lorsque l'on veut étudier un phénomène probabiliste, on considère un ensemble particulier Ω, contenant tous les résultats possibles des observations de ce phénomène.

Définition 2.1. *L'ensemble Ω contenant tous les résultats possibles des observations d'un phénomène est appelé* ensemble des observables.

Exemple 2.2. *Dans le cas d'un lancer de dé, les résultats possibles sont $\omega = 1, \omega = 2, ..., \omega = 6$ et l'ensemble des observables est $\Omega = \{1, 2, 3, 4, 5, 6\}$.* ♣

Tout sous-ensemble de l'ensemble des observables peut être considéré comme la représentation d'un *événement*. Dans le cas du lancer de dé, $A = \{1, 3, 5\}$ est l'événement correspondant à un résultat impair.

La théorie des probabilités consiste à associer aux événements leur *probabilité*. Dans le cas général, on n'assigne pas de probabilité à tous les événements possibles mais à une collection d'événements, notée $\mathcal{F}$ (où $\mathcal{F} \subseteq \mathcal{P}(\Omega)$).

Sur quel ensemble définir des probabilités ?

Ce que l'on souhaite, c'est avoir une collection dans laquelle on puisse tout d'abord définir la probabilité de Ω tout entier (qui sera 1). On veut de plus que la collection soit stable par complémentaire, pour que, lorsqu'on connaît la probabilité p d'un événement, on puisse calculer la probabilité du complémentaire ($= 1 - p$). On veut également pouvoir calculer la probabilité d'une union d'événements lorsqu'on connaît la probabilité de chacun.

Formellement, on définit la probabilité sur une collection notée $\mathcal{F} \subset \mathcal{P}(\Omega)$, satisfaisant la propriété de σ-algèbre suivante :

Définition 2.3. *Soit X un ensemble, et soit $\Sigma \subset \mathcal{P}(X)$ une famille de sous-ensembles. Σ est une* σ-algèbre *ou* tribu *si elle satisfait les propriétés suivantes :*

(1) $\emptyset \in \Sigma$

(2) $(A \in \Sigma) \Rightarrow (A^c \in \Sigma)$ *(où ^{c} dénote le complémentaire)*

(3) $(\forall i \in I, A_i \in \Sigma) \Rightarrow \bigcup_{i \in I} A_i \in \Sigma$ *pour tout ensemble I dénombrable.*

En exprimant l'intersection à l'aide des opérations d'union et de complémentaire, on déduit de cette définition que l'ensemble X appartient à Σ et qu'une σ-algèbre est fermée par intersection dénombrable. Bien évidemment, étant donné un ensemble d'observables Ω, il peut exister plusieurs σ-algèbres associées.

Exemple 2.4. *Considérons à nouveau l'exemple du lancer de dé. Imaginons que l'on ne s'intéresse qu'aux événements qui s'expriment en comparant le résultat du dé aux valeurs* 1 *et* 2*, et à l'aide d'opérateurs ensemblistes (intersection, union, complémentaire). On obtient alors la σ-algèbre suivante :*

$$\mathcal{F} = \{\{1\}, \{2\}, \{1,2\}, \{3,4,5,6\}, \{1,3,4,5,6\}, \{2,3,4,5,6\}, \emptyset, \Omega\}.$$

Ces événements correspondent respectivement à $\{1\}$, $\{2\}$, $\{1\} \cup \{2\}$, $(\{1\} \cup \{2\})^c$, $\{2\}^c$, $\{1\}^c$, $\{1\} \cap \{2\}$ *et* $(\{1\} \cap \{2\})^c$. *Dans ce cas, $\mathcal{F}$ est appelée* σ-algèbre engendrée *par les événements* 1 *et* 2. ♣

Lorsqu'un ensemble d'observables Ω est muni d'une σ-algèbre $\mathcal{F}$, on dit que $(\Omega, \mathcal{F})$ est un *espace mesurable*.

Cette notion de σ-algèbre $\mathcal{F}$ des ensembles mesurables sera utilisée à la section 2.2.2 puis à la section 2.3, lorsque l'on définira une notion de probabilité sur les chemins puis sur les exécutions. Pour des ensembles d'observables plus simples (comme par exemple la probabilité qu'une règle probabiliste donne un certain résultat), on prendra $\mathcal{F} = \mathcal{P}(\Omega)$.

Relativement à la notion d'espace mesurable, on peut définir ce qu'est une fonction mesurable :

Définition 2.5. *On appelle* fonction mesurable *d'un espace mesurable* (X, Σ) *dans un espace mesurable* (X', Σ') *toute fonction* $f : X \mapsto X'$ *telle que*

$$\forall E \in \Sigma',\ f^{-1}(E) \in \Sigma.$$

Dans la définition ci-dessus, la fonction f^{-1} désigne l'image réciproque par f, c'est-à-dire que $f^{-1}(E) = \{x \in X \mid f(x) \in E\}$. Une fonction mesurable assure donc que l'image réciproque de tout ensemble mesurable de Σ' est un ensemble mesurable appartenant à Σ.

La notion d'espace mesurable a été introduite dans le but d'y associer une fonction particulière, appelée mesure, qui associe une valeur à chaque ensemble mesurable. Nous allons considérer ici un cas particulier des mesures : celles de poids total 1, appelées mesures de probabilités.
La probabilité d'un événement mesure la fréquence de cet événement, la "chance" que l'on a de le constater lorsqu'on observe un phénomène probabiliste.

Définition 2.6. *On appelle* mesure de probabilité, *ou* distribution *sur un espace mesurable* $(\Omega, \mathcal{F})$ *toute application* $\mathbb{P}$ *de* $\mathcal{F}$ *dans* $\mathbb{R}^+$ *telle que :*

1. $\mathbb{P}$ *est définie pour tout élément de* $\mathcal{F}$,
2. $\mathbb{P}(\Omega) = 1$,
3. *pour tout ensemble dénombrable* $\{A_i, i \in I\}$ *d'éléments de* $\mathcal{F}$ *disjoints deux à deux, on a* $\mathbb{P}(\bigcup_{i \in I} A_i) = \sum_{i \in I} \mathbb{P}(A_i)$ *(σ-additivité).*

L'ensemble des distributions sur Ω *est noté* $\mathcal{D}(\Omega)$.
Le triplet $(\Omega, \mathcal{F}, \mathbb{P})$ *est appelé* espace de probabilités.

Exemple 2.7. *Pour le lancer de dé, et en prenant* $\mathcal{P}(\Omega)$ *comme σ-algèbre, pour tout ensemble* $A \subset \Omega = \{1, 2, 3, 4, 5, 6\}$, *la fonction*

$$\mathbb{P}(A) = \frac{|A|}{|\Omega|} = \frac{|A|}{6}$$

est une mesure de probabilité sur $(\Omega, \mathcal{P}(\Omega))$.

Toujours pour le lancer de dé mais avec la σ-algèbre $\mathcal{F}$ *donnée dans l'exemple 2.4, la restriction* $\mathbb{P}'$ *de* $\mathbb{P}$ *à* $\mathcal{F}$ *est une mesure de probabilité.* ♣

En général, lorsque l'on considère l'espace des réels, (resp. la droite $\overline{\mathbb{R}} = [-\infty, +\infty]$ des réels achevée) on lui associe une σ-algèbre particulière appelée σ-algèbre des boréliens, qui est la plus petite σ-algèbre engendrée[1] par les intervalles de la forme $]a, b]$ avec $-\infty \leq a < b < \infty$ (resp. avec $-\infty \leq a < b \leq \infty$). L'espace mesurable associé, dit de Borel, est noté $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ (resp. $(\overline{\mathbb{R}}, \mathcal{B}(\overline{\mathbb{R}}))$), ou plus simplement $(\mathbb{R}, \mathcal{B})$ (resp. $(\overline{\mathbb{R}}, \mathcal{B})$).

Définition 2.8. *Une* variable aléatoire *sur un espace de probabilité* $(\Omega, \mathcal{F}, \mathbb{P})$ *est une fonction mesurable* X *de* $(\Omega, \mathcal{F})$ *dans* $(\overline{\mathbb{R}}, \mathcal{B})$.

Une variable aléatoire sert à associer aux résultats observables $\omega \in \Omega$ une certaine valeur, fonction de ω, qui va nous intéresser.

[1] pour une preuve de l'existence d'une σ-algèbre engendrée par un ensemble, voir par exemple [KSK66]

Exemple 2.9. *Supposons que l'on s'intéresse toujours à un lancer de dés, mais cette fois ci dans le cadre d'un jeu, le but étant de faire le plus grand "score" avec les dés. On va alors associer à chaque résultat un score (par exemple la somme des valeurs des dés), et la fonction*

$$\begin{array}{rccc} X: & \Omega & \mapsto & \mathbb{N} \\ & (a,b,c) & \to & a+b+c \end{array}$$

est une variable aléatoire. ♣

Il reste encore à définir une notion qui sera cruciale pour la définition des chaînes de Markov, celle de *probabilité conditionnelle.*

Définition 2.10. *Soient* $(\Omega, \Sigma, \mathbb{P})$ *un ensemble de probabilités,* A *et* B *deux événements. On appelle* probabilité de A sachant B *et on note* $\mathbb{P}(A|B)$ *la probabilité définie (uniquement quand* $\mathbb{P}(B) > 0$*) par :*

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

Deux événements sont dits *indépendants* si la probabilité que A soit vrai ou non n'est pas influencée par le fait que B soit vrai. On a alors $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$ et $\mathbb{P}(A|B) = \mathbb{P}(A)$.

Exemple 2.11. *Toujours pour l'expérience de lancer du dé, considérons les trois événements suivants :*

A : ω est pair
B : ω est multiple de 3
C : ω est inférieur ou égal à 3

On a, pour la mesure de probabilité $\mathbb{P}$ *définie à l'exemple 2.7,* $\mathbb{P}(A) = 3/6$, $\mathbb{P}(B) = 2/6$ *et* $\mathbb{P}(C) = 3/6$. *Si on calcule les probabilités conditionnelles associées on a :*

$$\mathbb{P}(A|B) = \frac{1/6}{2/6} = \mathbb{P}(A), \quad \mathbb{P}(A|C) = \frac{1/6}{3/6} \neq \mathbb{P}(A), \text{ et } \mathbb{P}(B|C) = \frac{1/6}{3/6} = \mathbb{P}(B).$$

On constate donc que les événements A et B sont indépendants, de même que les événements B et C, mais pas les événements A et C.
De plus, B est indépendant des événements A et C pris séparément, mais clairement pas de l'événement $A \cap C$, car $\mathbb{P}(B|A \cap C) = 0$. ♣

A l'aide de toutes les notions présentées ci-dessus, nous sommes à présent en mesure d'introduire les deux modèles probabilistes que nous allons utiliser dans la suite : chaînes de Markov et processus de décisions markoviens. Mais avant cela, il est nécessaire d'insister sur la différence entre une action probabiliste et une action non déterministe. Il nous arrive souvent de penser que le choix probabiliste entre plusieurs états est un choix non déterministe. Seulement, par la suite nous aurons besoin d'une notion de non déterminisme plus forte, telle que le choix entre les états ne se fait pas de manière probabiliste mais de manière complètement arbitraire. C'est cette notion que nous appellerons non déterminisme à partir d'ici.

2.2 Chaînes de Markov

Les notions relatives aux chaînes de Markov présentées dans cette partie sont tirées des ouvrages [Bré99] et [Nor98].

2.2.1 Premières définitions

Soit S un ensemble dénombrable. Dans la suite on va appeler les éléments $s \in S$ des *états* et S l'*ensemble d'états*. En utilisant les notions de la section précédente, on peut définir ce qu'est une chaîne de Markov.

Définition 2.12. *Soit $(X_n)_{n\geq 0}$ une suite de variables aléatoires à valeurs dans S. $(X_n)_{n\geq 0}$ est une* chaîne de Markov *si $\mathbb{P}(X_n = j | X_0 = i_0, ..., X_{n-2} = i_{n-2}, X_{n-1} = i) = \mathbb{P}(X_n = j | X_{n-1} = i)$ pour tout $n > 0$ et tout n-uplet d'états $i_0, ..., i_{n-2}, i, j$ pour lequel les deux membres ci-dessus sont définis.*

Une chaîne de Markov est donc une suite de variables aléatoires telle que la distribution d'une variable ne dépend que de celle de la variable précédente, et pas de tout l'historique.

Une chaîne de Markov est dite *homogène* si $\mathbb{P}(X_n = j | X_{n-1} = i)$ ne dépend que de i et j, et pas de n. Dans ce cas on peut définir la *matrice de transition* $T = (t_{ij})_{i,j\in S}$ associée à la chaîne de Markov par $t_{ij} = \mathbb{P}(X_n = j | X_{n-1} = i)$. Cette matrice n'est pas quelconque : au contraire, elle vérifie que tout ses coefficients sont positifs, et que, pour chaque ligne, la somme des coefficients vaut 1 : $\forall i \in S, \sum_{j\in S} t_{ij} = 1$. Une telle matrice est appelée *matrice stochastique*.

Dans toute la suite on va toujours considérer, sans le préciser, des chaînes de Markov homogènes.

Pour se rapprocher du cadre informatique, on va considérer non plus une véritable chaîne de Markov, mais un *système markovien* à savoir un système dont le comportement suit celui d'une chaîne de Markov. A toute matrice de transition d'une chaîne de Markov correspond un graphe représentant un système markovien. Dans ce graphe, il existe une transition $s_i \xrightarrow{p} s_j$ de l'état s_i à l'état s_j étiquetée par $p > 0$ si et seulement si $t_{ij} = p$ dans la matrice de transition. Un exemple est donné en figure 2.1, correspondant à la matrice de transition suivante :

$$T = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/4 & 1/8 & 1/8 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 1/2 \end{pmatrix}$$

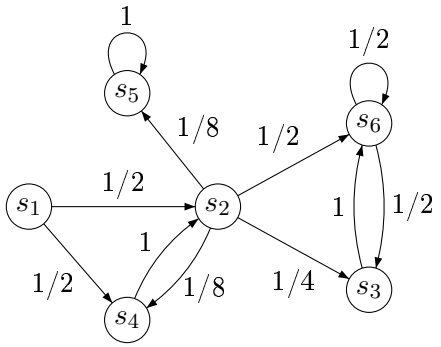


FIG. 2.1 – Un exemple de graphe associé à une chaîne de Markov

Un système markovien va alors être envisagé comme un système de transitions (*i.e.* un graphe orienté) probabiliste dans lequel, à chaque étape, on se déplace en tirant au sort le prochain état, et ce suivant les probabilités associées aux transitions.

Formellement, un *système de transitions probabiliste* est un graphe étiqueté tel que les étiquettes sont à valeurs dans $]0,1]$, et tel que pour tout état s du graphe, la somme des étiquettes des transitions partant de s vaut 1.

La représentation sous forme de graphe montre bien que, partant d'un certain état s_i, la probabilité d'aller en une étape dans un autre état s_j ne peut dépendre que de l'état s_i. Quand on arrive dans un état, on perd toute information sur comment on y est parvenu.

2.2.2 Mesure de probabilités sur l'ensemble des chemins

En se basant sur cette nouvelle caractérisation sous forme de systèmes de transitions probabilistes, on va pouvoir définir ce qu'est un chemin.

Définition 2.13. *Étant donné un système de transitions probabiliste G, un* chemin *(resp. un* chemin fini*) dans G est une suite infinie (resp. finie) d'états $s_0, s_1, ..., s_n, ...$ (resp. $s_0, s_1, ..., s_k$) telle que, pour tout i (resp. tout $i \in \{0, .., k-1\}$), (s_i, s_{i+1}) est une arête de G.*

Lorsqu'un système est markovien, la probabilité d'aller en une transition de l'état s à l'état s' est définie sans ambiguïté. On peut donc définir des probabilités sur les chemins.

Pour un chemin fini, on va prendre pour probabilité le produit des probabilités des transitions qui le composent, mais lorsque l'on a un chemin (infini), la définition de l'espace de probabilités nécessite plus de précautions.

Étant donnée une relation de transition probabiliste, on définit une relation d'accessibilité en un pas : $\rho \subset S \times S$ telle qu'on a $\rho(s,t)$ si et seulement si il existe une transition de probabilité p (forcément non nulle) entre s et t.

Pour tout état $s \in S$ on définit alors

$$\Omega_s = \{s_0 s_1 s_2 ... | s = s_0 \text{ et } \forall n \in \mathbb{N}, \rho(s_n, s_{n+1})\}$$

qui est l'ensemble de tous les chemins (infinis) issus de s.

Définition 2.14. *Étant donnée une suite finie $\sigma = s_0 = s, s_1, ..., s_k$ d'états de S on définit le* cylindre *de base σ et on note $\mathcal{C}_\sigma$ l'ensemble des chemins de Ω_s qui commencent par le chemin fini $s_0, s_1, ..., s_k$, autrement dit l'ensemble des chemins $s'_0, s'_1, s'_2, ..., s'_n, ...$ de Ω_s tels que $s'_1 = s_1, ..., s'_k = s_k$.*

Ici on n'a pas besoin de préciser $s'_0 = s_0$, car comme on prend des chemins de Ω_s, le premier état est fixé et vaut $s = s_0$. Un exemple de cylindre est donné à la figure 2.2.

Dans la littérature, on trouve également la notion de *cône* (cf. [BDLGJ02, SL95]) pour définir ces ensembles particuliers de chemins. Nous lui préférons celle de cylindre de [KSK66], voir aussi [BdA95].

Soit G un système de transitions associé à une chaîne de Markov de matrice de transition T. On définit la distribution $\mathbb{P}$ sur l'ensemble des cylindres de la façon suivante :

Soit $\mathcal{C}_\sigma$ un cylindre de base $\sigma = s_0, s_1, ..., s_k$. La probabilité $\mathbb{P}(\mathcal{C}_\sigma)$ est définie par :

$$\mathbb{P}(\mathcal{C}_{s_0,...,s_k}) = \prod_{0 \leq i \leq k-1} t_{s_i s_{i+1}}$$

.

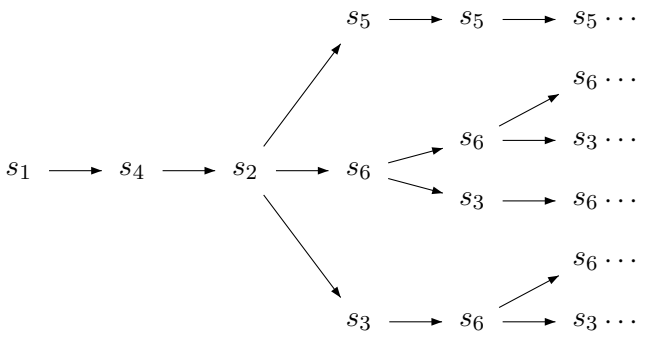


FIG. 2.2 – Ensemble des chemins d'un cylindre de base s_1, s_4, s_2

Exemple 2.15. *Reprenons le système donné à la figure 2.1. Le cylindre de base s_1, s_4, s_2 a comme probabilité $1/2$ $(= 1/2 \times 1)$ et désigne l'ensemble de chemins représenté à la figure 2.2.* ♣

On considère donc pour tout état s l'espace de probabilités $(\Omega_s, \mathcal{F}_s, \mathbb{P}_s)$ où $\mathcal{F}_s$ est la σ-algèbre engendrée par les cylindres dont la base est un chemin fini partant de s, et $\mathbb{P}_s$ est l'extension à $\mathcal{F}_s$ de la mesure de probabilités définie ci-dessus. Pour des définitions plus précises, entre autres de l'existence d'une σ-algèbre engendrée, voir par exemple [KSK66].

On a ainsi caractérisé les ensembles de chemins *mesurables*, c'est-à-dire ceux sur lesquels est définie la probabilité. Cette caractérisation est de plus pertinente car elle contient tous les ensembles de chemins définis par des formules de logique probabiliste que l'on voudra exprimer par la suite, comme par exemple l'ensemble des chemins qui passent par un ensemble donné, qui servira pour prouver la convergence.

2.2.3 États récurrents, états transitoires

A l'aide de notre définition de la mesure de probabilité sur les chemins, on peut définir ce qu'est un état récurrent.

Définition 2.16. *Un état s d'un système markovien est* récurrent *si la probabilité de revenir infiniment souvent dans cet état vaut* 1*, c'est à dire :*

$$\mathbb{P}_s(\{s_0, s_1, ..., s_n, ... \in \Omega_s | s_i = s \text{ pour une infinité d'indices } i\}) = 1.$$

Si ce n'est pas le cas, l'état est dit transitoire. *$\mathcal{R}ec$ est une notation pour l'ensemble des états récurrents.*

L'ensemble $\{s_0, s_1, ..., s_n, ... \in \Omega_s | s_i = s$ pour une infinité d'indices $i\}$ peut être décrit à l'aide d'unions et d'intersections dénombrables de cylindres, et la probabilité ci-dessus est bien définie.

Si on se restreint comme on va le faire dans les chapitres suivants au cas où l'ensemble d'états du système markovien est fini, alors on peut caractériser les états transitoires et récurrents de manière différente, en utilisant le graphe associé.

On va adopter la notation $s \to s'$ pour dire qu'il existe p strictement positif tel que $s \xrightarrow{p} s'$. On définit la fermeture transitive $\to^*$ de la relation $\to$ par

$$s \to^* s' \Leftrightarrow \exists s_0, ..., s_n \in S : \ s_0 = s, s_n = s' \text{ et } \forall i \in \{1, .., n-1\}, \ s_i \to s_{i+1}.$$

On a alors :

Proposition 2.17. *Pour un système markovien fini, un état s est récurrent ssi*

$$(i) \quad \forall s' \in S(s \rightarrow^* s' \Rightarrow s' \rightarrow^* s).$$

Il est transitoire ssi

$$(ii) \quad \exists s' \in S : \ (s \rightarrow^* s' \wedge \neg(s' \rightarrow^* s)).$$

La justification de cette classification des états d'un système markovien en états transitoires d'une part et états récurrents d'autre part sera donnée à la section 2.2.4 quand on aura défini la convergence probabiliste.

On peut constater que les deux propriétés (i) et (ii) ci-dessus forment une partition de l'ensemble des états. Cependant, dans le cas où l'ensemble d'états est infini dénombrable, la proposition ci-dessus ne peut pas s'appliquer, comme on peut le constater sur l'exemple figure 2.3. Tout état y satisfait (i) mais un calcul montre que tout état est transitoire : avec probabilité 1 on va diverger vers l'infini à droite.

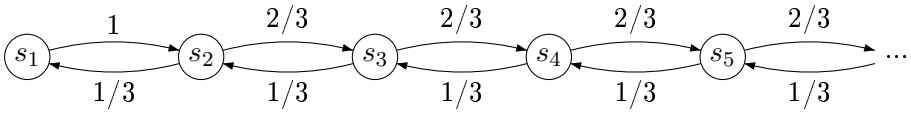


FIG. 2.3 – Une chaîne de Markov infinie sans états récurrents

Une autre caractéristique des états peut être importante dans le cadre de systèmes markoviens, c'est la notion d'états absorbants :

Définition 2.18. *Un état α d'une chaîne de Markov est* absorbant *si la probabilité de sortir de cet état est nulle, à savoir* $\mathbb{P}(X_{n+1} = \alpha | X_n = \alpha) = 1$. *L'ensemble des éléments absorbants est noté $\mathcal{A}bs$.*

En termes de système markovien, une fois qu'on arrive dans un état absorbant, on ne peut plus en sortir. Une chaîne de Markov est dite *absorbante* si tous les éléments récurrents sont absorbants et réciproquement ($\mathcal{R}ec = \mathcal{A}bs$).

On peut remarquer que, par définition, tout état absorbant est récurrent.

Si on reprend la caractérisation des chaînes de Markov par des graphes, on peut caractériser les états récurrents en termes de composantes fortement connexes. En effet, deux états s et s' d'un graphe appartiennent à la même *composante fortement connexe* si on a $s \rightarrow^* s'$ et $s' \rightarrow^* s$. L'ensemble des composantes fortement connexes d'un graphe forme une partition des sommets.

En reprenant la caractérisation des états récurrents donnée à la proposition 2.17, on constate que la propriété (i) est équivalente à :

Proposition 2.19. *Un état est récurrent si et seulement si il appartient à une composante fortement connexe fermée.*

Démonstration. Rappelons qu'un ensemble d'états est dit fermé si et seulement si la probabilité de sortir de cet ensemble est nulle.
Tout d'abord tout état appartient à une composante fortement connexe (que nous appellerons dans cette preuve CFC), et une seule. Il suffit donc de montrer que si elle est fermée l'état est

récurrent, et si elle est ouverte, l'état est transitoire. Supposons que s appartienne à une CFC fermée. On sait alors pour tout état s', si $s \to s'$ alors l'état s' appartient à la CFC de s, et donc $s' \to^* s$. Comme c'est vrai pour tout s', s est récurrent.
Supposons maintenant que la CFC de s ne soit pas fermée. Il existe donc s' appartenant à la CFC de s et s'' hors de la CFC de s tels que $s' \to s''$. Or par définition des CFC on a $s \to^* s'$ d'où $s \to^* s''$. Comme s'' n'est pas dans la CFC de s et qu'on vient de montrer que $s \to^* s''$, on ne peut pas avoir $s'' \to^* s$, ce qui prouve que s est transitoire. □

On peut également remarquer la particularité des états absorbants quant aux composantes fortement connexes : tout état absorbant est seul dans sa composante fortement connexe.

2.2.4 Temps moyen de convergence

Définition 2.20. *Étant donnés un système markovien sur un espace d'états S, et E un sous ensemble de S. On dit qu'il y a* convergence probabiliste *du système vers l'ensemble E si pour tout état s, la probabilité de l'ensemble des chemins issus de s qui passent par E vaut 1. Formellement cela s'écrit :*

$$\forall s \in S,\ \mathbb{P}_s(\sigma = s_0...s_n... \in \Omega_s | \exists i :\ s_i \in E) = 1.$$

On utilise le terme de convergence probabiliste, car ici on veut montrer que tous les chemins sauf un ensemble de probabilité nulle passent par l'ensemble E.

On peut remarquer, comme on l'avait annoncé avant, que l'ensemble des chemins caractérisé ci-dessus est bien dans $\mathcal{F}_s$ car c'est une union dénombrable d'unions finies de cylindres :

$$\bigcup_{i \in \mathbb{N}} \Big(\bigcup_{t_1 \in S} ... \bigcup_{t_{i-1} \in S} \bigcup_{t \in E} \{\sigma = s_0...s_n... \in \Omega_s | s_1 = t_1, ..., s_{i-1} = t_{i-1}, s_i = t\} \Big)$$

On remarque que la quantification ne porte pas sur s_0. Ceci est dû au fait que, comme on considère des chemins de Ω_s, s_0 est fixé et vaut s.

L'intérêt majeur de classifier les états en états récurrents d'une part, et états transitoires d'autre part réside dans le résultat suivant :

Théorème 2.21. (Markov) *Soit $\mathcal{S}$ un système se comportant comme une chaîne de Markov, on a convergence probabiliste de $\mathcal{S}$ vers l'ensemble $\mathcal{R}ec$ des états récurrents.*

Pour un état $s \in S$ donné, on a vu qu'on a un espace probabiliste $(\Omega_s, \mathcal{F}_s, \mathbb{P}_s)$. On a donc en particulier un ensemble mesurable $(\Omega_s, \mathcal{F}_s)$.
Étant donné un ensemble de configurations E, considérons la fonction X suivante, qui compte le nombre d'étapes le long d'un chemin nécessaires pour atteindre l'ensemble E :

$$\begin{array}{rccc} X : & \Omega_s & \mapsto & \overline{\mathbb{R}} \\ & \sigma = s_0 s_1 ... s_n ... & \to & i \text{ tel que } s_i \in E \wedge \forall j < i, s_j \notin E \end{array}$$

Cette fonction est une variable aléatoire. En effet, comme elle prend ses valeurs dans $\mathbb{N}$, pour tout intervalle $]a, b]$ de $\overline{\mathbb{R}}$ qui permet d'engendrer la σ-algèbre des boréliens (voir section 2.1) $X^{-1}(]a, b])$ est une union au plus dénombrable de $X^{-1}(c)$ où $c \in \mathbb{N} \cup \infty$. Or, $X^{-1}(\{c\})$ est une union finie de cylindres, dont la base est de longueur c, dont les $c-1$ premiers états ne sont pas dans E mais le $c^{\text{ème}}$ est dans E.

En ce basant sur les considérations ci-dessus, le *temps moyen de convergence* d'un système markovien se calcule en faisant une "moyenne" sur l'ensemble des chemins de la longueur de ces chemins (la valeur de $X(\omega)$), cette moyenne étant pondérée par la probabilité de ces chemins. Cette moyenne, appelée aussi *espérance* de la variable aléatoire X, peut se calculer de la façon suivante :

$$E(X) = \sum_{k \in \mathbb{N} \cup \infty} k \times \mathbb{P}(X^{-1}(\{k\}))$$

Bien sûr, pour que cette somme soit finie il faut que l'ensemble des chemins de $X^{-1}(\{\infty\})$ soit de probabilité nulle, la condition ci-dessus ($\mathbb{P}(X^{-1}(\{\infty\})) = 0$) étant nécessaire mais pas suffisante.

2.3 Processus de décision markoviens

Les résultats et définitions que nous allons donnés sont tirés de [dA97a] et de [Put94]. Un processus de décision markovien est une généralisation de la notion de chaîne de Markov dans lequel un ensemble d'actions possibles est associé à chaque état. A chaque couple (état,action) correspond une distribution de probabilité, qui est utilisée pour déterminer l'état successeur [Der70, Ber95]. Une chaîne de Markov correspond ainsi à un processus de décision markovien dans lequel une seule action est associée à chaque état. Par la suite, nous supposerons l'existence d'un ensemble *Acts* d'actions, contenant toutes les actions en jeu. la définition d'un processus de décision markovien est comme suit :

Définition 2.22. (Processus de décision markovien) *Un processus de décision markovien est un triplet (S, A, p) tel que :*

- *S est un ensemble fini d'états*
- *Pour tout $s \in S$, $A(s) \subseteq Acts$ est un ensemble non vide d'actions disponibles dans l'état s.*
- *Pour tout $s, t \in S$ et $a \in A(s)$, $p_{st}(a)$ est la probabilité de la transition de s vers t quand l'action a a été sélectionnée. Pour tout $s, t \in S$ et $a \in A(s)$, on a $0 \leq p_{st}(a) \leq 1$ et $\sum_{t \in S} p_{st}(a) = 1$.*

A partir d'ici nous ne considérerons uniquement les processus de décision markoviens finis, c'est à dire, ayant un ensemble fini d'états et d'actions.

Exemple 2.23. *(Un jeu de dés : le 421) Ce jeu est un jeu très connu des adeptes des comptoirs de cafés et peut être facilement étudié comme un processus de décision markovien.*
Ce jeu se joue avec trois dés et le but du jeu est d'obtenir une meilleure combinaison que ses adversaires. Voici l'ordre des combinaisons pour chaque joueur :

1. 421
2. 111
3. 611
4. 666
5. 511
6. 555

7. 411
8. 444
9. 311
10. 333
11. 211
12. 222
13. 654
14. 543
15. 432
16. 321

Puis par ordre décroissant de valeur depuis 665 *à* 221.
Nous noterons toujours la combinaison dans l'ordre décroissant et nous appellerons dé 1 *le dé avec la plus grande valeur dé* 2 *celui de valeur intermédiaire et dé* 3 *celui de plus petite valeur.*
Le jeu se déroule ainsi :
Le premier joueur lance les trois dés puis décide soit :

- *de s'Arrêter (action A)*
- *de relancer Un dé (actions U_1, U_2, et U_3)*
- *de relancer Deux dés (actions D_{12}, D_{13}, et D_{23})*
- *ou de relancer les Trois dés (action T)*

Il fait son choix, relance les dés (s'il n'a pas choisi l'action A) puis recommence à nouveau (en tout il peut y avoir trois lancers de dés que l'on appellera des essais). On note son résultat puis c'est au deuxième joueur de jouer.
Le joueur ayant réussi la meilleure combinaison a gagné. Dans cet exemple nous avons :

- $Acts = \{A, U_1, U_2, U_3, D_{12}, D_{13}, D_{23}, T\}$
- $S = \{(abc, n) | a \geq b \geq c \wedge (a, b, c) \in \{1, 2, 3, 4, 5, 6\}^3 \wedge n \in \{1, 2, 3\}\}$ *où abc représente la configuration des trois dés et n représente le nombre d'essais déjà effectués.*
- *Si* $s = (abc, 3)$ *alors* $A(s) = \{A\}$ *sinon* $A(s) = \{A, U_1, U_2, U_3, D_{12}, D_{13}, D_{23}, T\}$

Voici maintenant deux lois de probabilités possibles dépendant de l'action choisie et partant de la configuration $(211, 1)$. *Dans la figure 2.4 le graphe de gauche représente les transitions si le choix* U_1 *est fait, celui de droite pour le choix* U_2

♣

2.3.1 Exécutions

Une exécution d'un processus de décision markovien est une suite infinie composée d'états et d'actions alternés construite suivant un processus à deux phases. Premièrement, étant donné un état s, une action a est sélectionnée de manière non déterministe puis un successeur t à s est choisi suivant la distribution $\mathbb{P}(t|s, a) = p_{st}(a)$. Voici la définition formelle d'une exécution.

Définition 2.24. (Exécution d'un processus de décision markovien) *Une exécution d'un processus de décision markovien* Π *est une suite infinie* $\omega = s_0 a_0 s_1 a_1 ...$ *telle que* $s_i \in S$, $a_i \in A(s_i)$ *et* $p_{s_i, s_{i+1}}(a_i) > 0$ *pour tout* $i \geq 0$. *De la même manière que pour les chaînes de Markov, étant donné un état s, nous noterons* Ω_s *l'ensemble des exécutions partant de s. Nous noterons aussi* X_i *(resp.* Y_i*) la variable aléatoire représentant le* $i^{ème}$ *état (resp. action) du processus de décision markovien.*

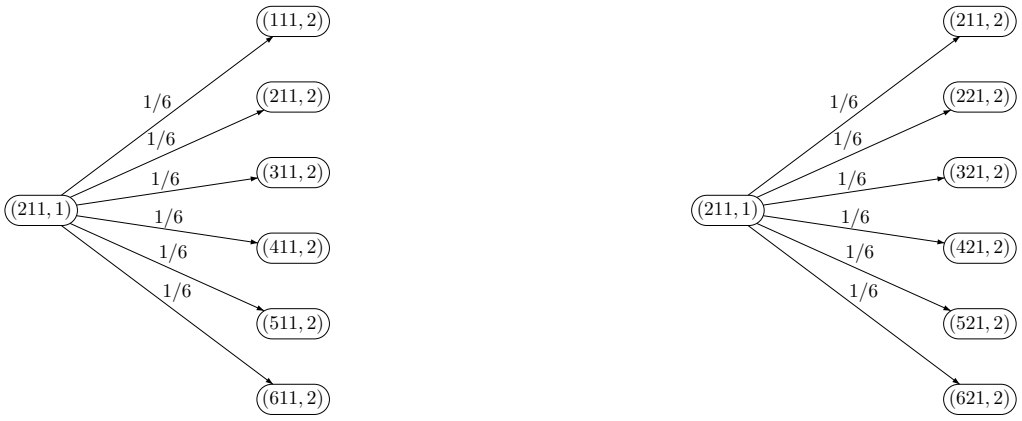


FIG. 2.4 – Transitions suivant le choix de l'action

Exemple 2.25. *Nous allons prendre ici une variante de notre jeu* 421. *Vous jouez en second contre un ordinateur et votre but est de faire mieux que lui le plus vite possible (i.e. en peu d'essais). Nous utiliserons à présent cette variante jusqu'à la fin de l'exposé. Une exécution possible est :*

$$532, D_{12}, 442, U_1, 432, U_2, ...$$

Ici la deuxième composante de l'état n'est pas nécessaire (on joue autant d'essais que l'on veut). ♣

Nous allons définir maintenant la notion de couple $état - action$ en suivant les notations de [dA97a].

Définition 2.26. (couples $état - action$**)** *Un couple* $état - action$ *est un couple* s, a *tel que* $s \in S$, $a \in A(s)$. *On notera* $\chi_\Pi = \{(s, a) | s \in S \wedge a \in A(s)\}$ *l'ensemble des paires* $état - action$ *d'un processus de décision markovien* Π. *Pour tout paire d'*$état - action$ s, a, *on définira* $Succ(s, a) = t | p_{st}(a) > 0$ *comme l'ensemble des successeurs possibles de* s *lorsque* a *est choisi.*

A l'instar des définitions faites dans le cadre des chaînes de Markov, nous allons définir à présent les ensembles mesurables d'exécutions.

Ensembles mesurables d'exécutions

Pour tout état $s \in S$, soit $\mathcal{B}_s \subseteq 2^{\Omega_s}$ la plus petite algèbre de sous ensemble de Ω_s qui contient tous les cylindres de bases :

$$\{\omega \in \Omega_s | X_0 = s_0 = s \wedge Y_0 = a_0 \wedge ... \wedge X_n = s_n \wedge Y_n = a_n\}$$

pour tout $n \geq 0$, $s_0, ...s_n \in S$, $a_0 \in A(s_0)$,...,$a_n \in A(s_n)$, et qui est close par complémentaire, et par unions et intersections dénombrables. Cette algèbre est appelée la σ-algèbre de Borel, ces éléments sont les ensembles mesurables d'exécutions, sur lesquels il est possible de calculer une probabilité, voir aussi [KSK66, Wil91].

2.3.2 Politiques

Il existe plusieurs notions de politiques (que nous discuterons dans ce paragraphe). Nous avons choisi de nous fixer ici sur celle définie dans [dA97a], dont nous redonnons les principales propriétés.
Pour pouvoir définir des probabilités sur les exécutions, il faut associer à chaque $\Delta \in \mathcal{B}_f$ sa probabilité $\mathbb{P}(\Delta)$. Mais, cette mesure de probabilité n'est pas bien définie, en effet la probabilité qu'une exécution ω appartienne à Δ dépend de la manière dont les choix non déterministes ont été faits.
Pour modéliser ces choix, nous utiliserons le concept de *politique* (voir Derman [Der70]). Ce concept est très voisin de ceux d'adversaires de Segala et Lynch [SL95] et d'ordonnancement (scheduler) de Lehman et Rabin [LR81], Vardi [Var85] et Pnueli et Zuck [PZ86].

Définition 2.27. (Politique) *Une politique η est un ensemble de probabilités conditionnelles $Q_\eta(a|s_0s_1...s_n)$, définies pour tout $n \geq 0$, pour toute séquence d'états $s_0...s_n$ et pour tout $a \in A(s_n)$ et tel que l'on ait :*

$$0 \leq Q_\eta(a|s_0s_1...s_n) \leq 1 \qquad \sum_{a \in A(s_n)} Q_\eta(a|s_0s_1...s_n) = 1.$$

De manière informelle, une politique édicte la manière par laquelle les choix non déterministes vont être pris. En effet, après un début d'exécution $s_0s_1...s_n$, l'action a est choisie avec la probabilité $Q_\eta(a|s_0s_1...s_n)$.
Ainsi, sous une politique η, la probabilité d'une transition vers t après $s_0s_1...s_n$ que l'on notera $\mathbb{P}^\eta_{s_0}(t|s_0s_1...s_n)$ est :

$$\sum_{a \in A(s_n)} p_{s_n,t}(a) Q_\eta(a|s_0s_1...s_n).$$

pour finir remarquons que la probabilité d'un début d'exécution $s_0a_0s_1a_1...s_n$ sous la politique η est donnée par :

$$\prod_{i=0}^{n-1} p_{s_i,s_{i+1}}(a_i) Q_\eta(a_i|s_0s_1...s_n).$$

Ces définitions sur les exécutions finies permettent de définir une mesure de probabilité $\mathbb{P}^\eta_s$ sur $\mathcal{B}_s$. Pour plus de détails concernant l'extension de la mesure sur la σ-algèbre à partir des cylindres de base, on pourra se reporter à [Doo94], [KSK66] ou [Wil91].

Exemple 2.28. *Revenons à notre jeu de dés préféré (avec un nombre d'essais infini) et définissons deux politiques différentes :*

- *La politique tout pour le 421 : η_{421}. Elle consiste à retirer tous les dés qui ne permettent pas de faire* 421 *(par exemple si on a* 631 *on retire deux dés, et si on a* 411 *on en retire un). Bien sur si votre combinaison est meilleure que celle de l'ordinateur vous vous arrêtez.*
- *La politique tout pour les triples : η_{aaa}. Elle consiste à viser les triples meilleurs que la combinaison de l'ordinateur. Notons que si l'ordinateur a réussi* 111 *ou* 421 *cette politique ne permet pas de gagner. Mais, si par exemple l'ordinateur a tiré* 211 *et que votre combinaison est* 321 *vous choisissez l'action D_{12} avec probabilité* 1/2 *(pour faire* 111*), l'action D_{23} avec probabilité* 1/2 *(pour faire* 333*).*

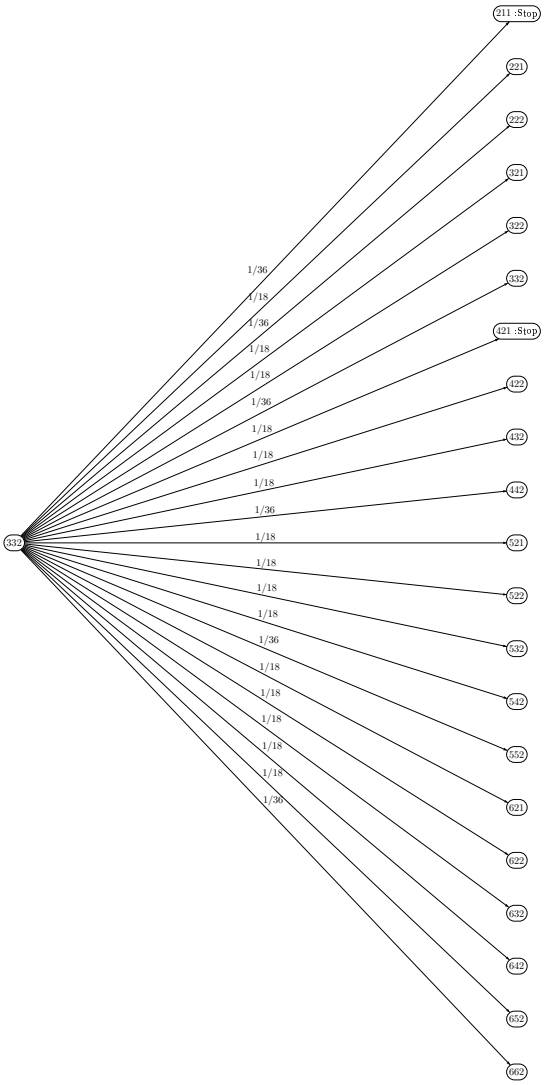


FIG. 2.5 – Evolution sous la politique η_{421}

Regardons des débuts d'exécutions possibles suivant les deux stratégies sachant que l'ordinateur a réussi 222 *et que notre premier essai était* 332.
Dans la figure 2.5, l'évolution sous la politique μ_{421} *est décrite, la probabilité de s'arrêter est donc de* $1/12$.*Alors que dans la figure 2.6, on décrit celle suivant la politique* μ_{aaa}*, ici la probabilité de s'arrêter est de* $1/6$.

Les stratégies offrent donc des résultats différents et nous étudierons par la suite la notion de stratégie optimale. ♣

Nous allons définir, à présent la notion de **politique markovienne et déterministe** . Une politique est dite markovienne, si elle ne dépend que du dernier état atteint dans l'exécution, on dit aussi que cette politique est sans mémoire, plus formellement :

Définition 2.29. (Politiques markoviennes et déterministes) *Une politique* η *est dite*

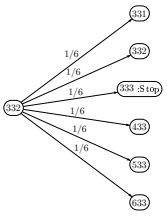


FIG. 2.6 – Evolution sous la politique η_{aaa}

markovienne si :

$$Q_\eta(a|s_0 s_1 ... s_n) = Q_\eta(a|s_n)$$

pour tout $n \geq 0$ *et tout suite* $s_0 ... s_n$ *d'états de* S. *toujours en suivant les notations de [dA97a], nous noterons* η *l'ensemble de toutes les politiques et* $\eta_M \subset \eta$ *celles qui sont markoviennes. Une politique* η *est dite déterministe si elle est markovienne, et si pour chaque* s *il y a une action* $a \in A(s)$ *telle que* $Q_\eta(a|s) = 1$. *Nous noterons* η_D *l'ensemble des politiques déterministes.*

Exemple 2.30. *On peut remarquer que la politique* η_{421} *est déterministe car l'action est exactement fixée par la configuration courante en rejouant les dés qui ne permettent pas d'obtenir* 421. *Par contre, la politique* η_{aaa} *est markovienne mais non déterministe car parfois on peut avoir un choix probabiliste entre plusieurs actions.* ♣

Digression sur la notion de politique

La définition de politique que nous avons retenue, n'est pas la plus générale possible, en effet on aurait pu remplacer la définition par

$$Q_\eta(a|s_0 a_0 s_1 a_1 ... s_n),$$

en supposant que le choix de la prochaine action puisse aussi dépendre de l'historique des actions passées. Mais nous ne l'avons pas fait pour être conformes à la littérature de référence [Der70] [Put94]. Ainsi, nous pourrons utiliser les résultats prouvés par ces auteurs. De plus, les résultats de model checking établis notamment par De Alfaro sont énoncés avec ce formalisme, et il est facile de prouver qu'ils sont aussi valables en prenant la notion de politique étendue.
Une deuxième comparaison possible est celle avec les politiques qui se basent sur les **règles de décisions**. Cette approche est assez courante dans la littérature sur les processus de décision markoviens (notamment dans [Ber95]). On peut la définir ainsi :
Une règle de décision δ est une fonction qui associe à chaque état s une distribution de probabilité $\{\delta_s(a)\}_{a \in A(s)}$ sur les actions associées à s. une politique est donc définie par une suite infinie de règles de décisions $\delta_1 ... \delta_n ...$, après avoir suivie une exécution $s_0 ... s_n$ la règle de dècision δ_n nous permet de choisir la prochaine action.
Cette notion est plus restrictive que celle que nous avons choisie car le choix de l'action ne dépend que de la longueur de l'exécution mais pas du passé. Elle est utilisée car elle simplifie beaucoup de choses notamment dans l'étude matricielle des processus de décision markoviens. Mais, suivant de Alfaro, nous avons fait le choix de la définition la plus générale.

2.3.3 Représentation graphique des processus de décision markoviens

Il est souvent utile pour décrire un processus de décision markovien $\Pi = (S, A, p)$ d'avoir une représentation graphique adaptée. Nous allons décrire ici celle qui est le plus souvent utilisée. La représentation est celle d'un graphe dans lequel les noeuds sont les éléments $s \in S$. Pour tout $s \in S$ et $a \in A(s)$ nous dessinons un faisceau d'arcs pour chaque t tel que $p_{st}(a) > 0$. Les arcs appartenant au même faisceau sont reliés ensemble par un petit arc de cercle, et chaque faisceau porte le label de l'action correspondante. Voici, par exemple, une représentation graphique d'un processus de décision markovien simple (voir 2.7).

Exemple 2.31. Le barman aveugle

Cet exemple classique de théorie des automates va nous permettre d'illustrer la manière dont on représente graphiquement les processus de décision markoviens. Un barman a les yeux bandés et il y a quatre jetons sur son plateau disposés pour former un carré (voir figure 2.7 (une face des jetons est blanche l'autre face est noire). A chaque coup, il décide s'il retourne un jeton (action Un), deux jetons adjacents (action Adj) ou deux jetons opposés (action Opp), puis tire au sort aléatoirement le (ou les jetons qu'il retourne). Son but, est que les jetons soient tous retournés dans le même sens à la fin. Voici la représentation graphique du processus de décision markovien associé dans lequel chaque état est la classe des configurations à inversion de couleur près et à rotation près :

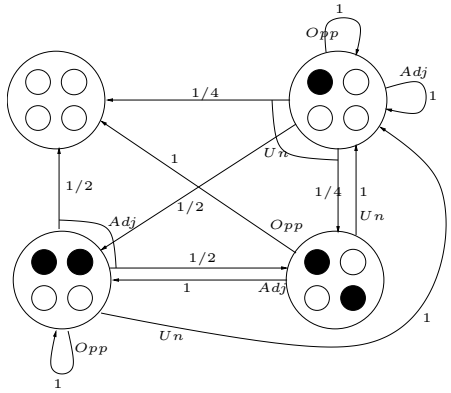


FIG. 2.7 – représentation graphique du problème du barman aveugle

Nous pouvons détailler ici l'ensemble des caractéristiques de ce processus de décision markovien :

- $S = \{s_1, s_2, s_3, s_4\}$ *où :*
  - s_1 *est l'état qui représente les configurations dans lesquelles les quatre jetons sont de la même couleur.*
  - s_2 *celui qui représente les configurations avec un jeton d'une couleur différente des trois autres.*
  - s_3 *celui qui représente les configurations avec les deux jetons opposés de la même couleur.*
  - s_4 *qui représente les configurations avec deux jetons adjacents de la même couleur.*
- *Pour tout* $s \in \{s_2, s_3, s_4\} A(s) = \{Un, Opp, Adj\}$, $A(s_1) = \emptyset$.
- *Et les différentes lois de probabilités :*

$t =$	s_1	s_2	s_3	s_4
$p_{s_2,t}(Un)$	1/4	0	1/4	1/2
$p_{s_2,t}(Opp)$	0	1	0	0
$p_{s_2,t}(Adj)$	0	1	0	0
$p_{s_3,t}(Un)$	0	1	0	0
$p_{s_3,t}(Opp)$	1	0	0	0
$p_{s_3,t}(Adj)$	0	0	0	1
$p_{s_4,t}(Un)$	0	1	0	0
$p_{s_4,t}(Opp)$	0	0	0	1
$p_{s_4,t}(Adj)$	1/2	0	1/2	0

♣

2.3.4 Accessibilité

Après avoir donné les définitions basiques qui nous serviront à manipuler les processus de décision markoviens, nous allons dans cette partie décrire les problématiques sur lesquelles nous nous sommes penchés. Elle consiste en deux principales quantités qui sont la probabilité d'atteindre un ensemble donné et le temps (moyen, maximal, minimal) d'atteinte.

Définitions

Il est nécessaire de donner, au préalable, quelques notations.

Définition 2.32. (Accessibilité) *Etant donné un processus de décision markovien (S, A, p) on dira qu'un état t est accessible depuis s s'il existe un chemin de s à t dans le graphe (S, ρ_S). On dit qu'un ensemble $T \subseteq S$ est accessible depuis $U \subseteq S$ s'il existe deux états $t \in T$ et $u \in U$ tels que t est accessible depuis u. On peut ainsi définir les notions d'accessibilité depuis un état vers un ensemble d'états (et vice versa).*

On peut aussi étendre la notion de forte connexité aux processus de décision markoviens, en effet un processus de décision markovien est fortement connexe si tous les états sont accessibles depuis tous les états (i.e : (S, ρ_S) est fortement connexe).

Définition 2.33. (Evénement reach) *Soit un sous ensemble T d'états, on définit l'événement $reach(T)$ par $\exists k X_k \in T$. Cet événement est donc vrai sur une exécution si cette exécution atteint T*

Enfin, nous aurons besoin de définir une fonction aléatoire :

Définition 2.34. ($inft$) *Etant donnée une exécution ω, on définit :*

$$inft(\omega) = \{(s, a) | \forall n \exists k > n X_k = s \wedge Y_k = a\}$$

qui représente l'ensemble des couples états-actions qui apparaissent une infinité de fois dans ω.

Convergence

Dans cette partie nous allons décrire les propriétés stochastiques des processus de décision markoviens. Nous le ferons de la même manière que pour les chaînes de Markov et toujours en suivant le formalisme de [dA97a].

Définition 2.35. (Ensemble d'état-action et sous-PDM) *Soit un processus de décision markovien $\Pi = (S, A, p)$, un ensemble état-action est un sous-ensemble $\chi \subseteq \{(s,a)|s \in S \wedge a \in A(s)\}$. Un sous-PDM de Π est un couple (C, D) où $C \subseteq S$ et D est une fonction qui associe à chaque $s \in C$ un ensemble $D(s) \subseteq A(s)$ d'actions. On peut donc facilement établir une correspondance entre les ensembles d'état-action et les sous-PDM définie par :*

- *Etant donné un ensemble état-action χ, on obtient le sous-PDM $sub(\chi) = (C < D)$ avec :*
$$C = \{s|\exists a(s,a) \in \chi\} \qquad D(s) = \{a|(s,a) \in \chi\}$$
- *Etant donné un sous-PDM (C,D), on obtient un ensemble d'état-action $sa(C, D) = \{(s,a)|s \in C \wedge a \in D(s)\}$*

Il est évident que chaque sous-PDM induit une relation binaire (arcs) sur la représentation graphique du processus de décision markovien. En effet on dira qu'il y a une arc de $s \in C$ vers $t \in S$ (notons qu'a priori rien ne dit que $t \in C$) si et seulement si il est possible d'aller de s à t avec une probabilité positive en choisissant une action de $D(s)$; nous noterons $\rho_{(C,D)}$ l'ensemble des arcs induits par ce sous-PDM.

Nous pouvons à présent définir le pendant des ensembles d'états récurrents d'une chaîne de Markov pour les processus de décision markoviens. Toujours en suivant le formalisme de [dA97a], nous l'appellerons **composants finaux**.

Définition 2.36. (composants finaux) *Un sous-PDM (C, D) est un composant final si :*
- *$Succ(s, a) \subseteq C$ pour tout $s \in C$ et $a \in D(s)$;*
- *Le graphe $(C, \rho_{(C,D)})$ est fortement connexe.*

De manière informelle, on peut remarquer que les composants finaux représentent les ensemble d'état-action tels qu'une fois atteints, et suivant une politique appropriée on y reste. Donnons quelques exemples.

Exemple 2.37. *Reprenons encore notre exemple du 421, on pourrait définir un composant final (parmi d'autres) (C, D) par :*
- *$C = \{s \in S|s = 66x\}$, soit toutes les configurations possibles avec deux 6.*
- *Pour tout $s \in C$ on pose $D(s) = U_3$*

il est évident de montrer que sous cette politique le graphe $(C, \rho_{(C,D)})$ est fortement connexe, et que tous les successeurs des éléments de C sont dans C. ♣

Exemple 2.38. (Le barman aveugle) *Dans cet exemple, on peut aussi trouver plusieurs composants finaux. par exemple (C, D) tel que :*
- *$C = \{s_2\}$*
- *$D(s_2) = \{Un, Adj\}$.*

La preuve est évidente. ♣

La description des théorèmes principaux sur le comportement stochastique des processus de décision markoviens va nous permettre de mieux comprendre cette notion.

Notre premier théorème décrit la stabilité des composants finaux. En effet une fois un composant final atteint il existe une politique permettant qu'on y reste et que les états visités infiniment souvent soient ceux du composant.

Théorème 2.39. (stabilité des composants finaux) *Soit (C, D) un composant final. Pour toute politique η, il existe une politique η', qui diffère de η seulement sur C, telle que :*

$$\mathbb{P}_s^{\eta}(reach(C)) = \mathbb{P}_s^{\eta'}(reach(C)) = \mathbb{P}_s^{\eta'}(inft(\omega) = sa(C, D))$$

pour tout $s \in S$.

Démonstration. La démonstration de ce théorème est donnée dans [dA97a] et dans [dA97b]. Mais sa relative simplicité nous permet de la donner ici.

On définit la politique η' , pour tout suite finie $s_0, s_1, ...s_n$:

– Si $s_n \in C$, la politique η' choisit uniformément une action de $D(s_n)$

– Si $s_n \notin C$, la politique η' coïncide avec η.

La première égalité est la conséquence du fait que η et η' coïncident en dehors de C. La deuxième égalité vient du fait que sous la politique η' une exécution qui entre dans C n'en sort plus. De plus, comme le graphe $(C, \rho_{(C,D)})$ est fortement connexe, sous la politique η' la composante finale se comporte exactement comme une chaîne de Markov irréductible récurrente. Ainsi tous les états de C seront visités infiniment souvent. □

Le deuxième théorème que nous donnons ici, exprime le fait que pour un état initial et une politique donnée une exécution va terminer avec probabilité 1 dans un composant final, d'où son nom.

Théorème 2.40. (Théorème fondamental des composants finaux) *Pour tout $s \in S$ et pour tout politique η :*

$$\mathbb{P}_s^{\eta}(sub(inft(\omega)) \text{ est un composant final}) = 1.$$

La preuve de ce théorème est donnée en page 46 de [dA97a]. Remarquons juste que si la politique choisie est markovienne, il est évident de voir que le processus de décision markovien devient une chaîne de Markov et que le résultat énoncé ici n'est autre que celui énoncé dans le paragraphe sur les chaînes de Markov (théorème 2.21).

Propriétés

Dans cette dernière partie, nous allons définir les deux principaux problèmes que nous allons être amené à vérifier par la suite. Les algorithmes pour résoudre ces problèmes seront donnés dans les chapitres suivant, nous nous contenterons ici d'en définir les termes.
Le premier est le problème de la probabilité d'accessibilité maximale. Ce problème consiste à trouver la probabilité maximale pour atteindre un ensemble d'états donné à partir d'un état initial lui aussi connu. En voici une définition formelle.

Définition 2.41. (Instance du problème de la probabilité d'accessibilité maximale) *Une instance du problème de la probabilité d'accessibilité maximale consiste en un PDM $\Pi = (S, A, p, U)$ où $U \subseteq S$ représente l'ensemble des destinations.*

Définition 2.42. (Problème de la probabilité d'accessibilité maximale) *Etant donnée une instance $\Pi = (S, A, p, U)$, le problème d'accessibilité maximale consiste à calculer la quantité*

$$v_s^* = \sup_{\eta} \mathbb{P}_s^{\eta}(reach(U))$$

pour tout $s \in S - U$.

Nous pouvons définir de la même manière un problème analogue qui nous intéressera par la suite : le problème de la probabilité d'accessibilité minimale. Il consiste à trouver la probabilité minimale pour atteindre un certain ensemble d'états.

Exemple 2.43. *Dans le cas du jeu* 421*, supposons que l'on joue seul cette fois et que notre unique but est de faire une des seize configurations gagnantes. Le problème qu'on se pose est de savoir si quelle que soit la politique on arrive quand même dans une des seize configurations gagnantes (ensemble G) avec probabilité non nulle. C'est à dire que le problème de la probabilité d'accessibilité minimale de G a comme solution* 1 *ou non.* ♣

Le deuxième problème sur lequel nous nous pencherons est le problème du pire temps moyen d'atteinte. Ce problème n'apparaît que si on a montré (à l'aide de la solution du problème précédent) qu'un sous ensemble d'états (forcément un composant final, qu'on appellera ensemble ergodique pour le PDM) était atteint avec probabilité 1 quelle que soit la politique choisie. Dans ce cas, voici les définitions formelles.

Définition 2.44. (Instance du problème du pire temps moyen d'atteinte) *Une instance du problème du pire temps moyen d'atteinte est donnée par un PDM* $\Pi = (S, A, p, U)$ *dans lequel* U *est un ensemble ergodique pour* Π

Avant de donner la définition formelle de ce problème il est nécessaire de donner une définition du temps moyen de convergence pour une politique donnée. Notons tout d'abord que si la politique est markovienne la définition est donnée dans le paragraphe 2.2.4, car, alors le processus de décision markovien est une chaîne de Markov.

Définition 2.45. (Temps moyen de convergence) *Etant données une instance du problème du pire temps moyen d'atteinte, et une politique* η*, nous appellerons temps moyen de convergence pour la politique* η *la quantité* $E(X_\eta)$ *où la variable aléatoire* X_η *est telle que :*

$$\begin{array}{rccc} X_\eta : & \Omega_s^\eta & \mapsto & \overline{\mathbb{R}} \\ & \omega = s_0 s_1 ... s_n ... & \rightarrow & i \text{ tel que } s_i \in U \wedge \forall j < i, s_j \notin U \end{array}$$

dans lequel ω *est une exécution possible sous la politique* η*.*

Définition 2.46. (Problème du pire temps moyen d'atteinte) *Etant donnée une instance du problème du pire temps moyen d'atteinte, le problème consiste à calculer la quantité*

$$\sup_{\eta \in Pol} E(X_\eta)$$

Exemple 2.47. *Si on reprend encore une fois le jeu du* 421 *résoudre ce problème nous permettrait de savoir en combien de temps au pire on arrive à une des seize configurations gagnantes (G) même si on n'essaie pas de le faire (avec la pire stratégie).* ♣

Chapitre 3

Problème du plus court chemin stochastique et applications

Le plus court chemin entre deux points est la ligne droite à condition que les deux points soient bien en face l'un de l'autre.
Pierre Dac

Dans ce chapitre, nous allons donner une manière (algorithmique) pour résoudre les problèmes évoqués à la fin du chapitre 2. En effet on peut remarquer que ces trois problèmes (Probabilité d'accessibilité maximale ou minimale et pire temps moyen d'atteinte) sont des instances d'un problème générique : le problème du plus court chemin aléatoire (SSP pour Stochastic Shortest Path). Nous adopterons pour le décrire les formalismes de [dA97a], [dA99] et de [BT91].

3.1 Définition du problème

Le problème SSP est la version probabilité du problème du plus court chemin. Afin de le rendre le plus général possible, on peut affecter des poids (appelés coûts) aux transitions et aux états. De même, on doit définir l'ensemble cible R. Ainsi, résoudre une instance du problème SSP consistera à trouver le plus petit coût possible (en espérance) sur toutes les politiques qui garantissent l'accès à l'ensemble cible R. Voici maintenant quelques définitions formelles (suivant les notations de [dA99]) pour définir le problème.

Définition 3.1. Fonction de coût des transitions
$c : S \times Acts \to \mathbb{R}$ *est appelée fonction de coût des transitions, elle associe à chaque état* $s \in S - R$ *et à chaque action* $a \in A(s)$ *le coût* $c(s, a)$.

Définition 3.2. Fonction de coût terminale
$g : R \to \mathbb{R}$ *est appelée fonction de coût terminale, elle associe à chaque* $s \in R$ *son coût terminal* $g(s)$.

Exemple 3.3. *Si nous reprenons notre exemple du jeu de dés* 421 *on peut définir l'ensemble cible comme l'ensemble des configurations meilleures que celle faite par l'adversaire. Pour nous faciliter la compréhension, supposons que l'adversaire (une machine par exemple) ait réussi la combinaison* 555. *L'ensemble cible est donc* $R = \{421, 111, 611, 666, 511\}$. *A présent, nous*

pouvons définir les coûts associés aux transitions, de la manière la plus simple possible, chaque nouveau lancer de dés nous coûte 10 *ainsi* $\forall s \in S - R, \forall a \in A(s), c(s,a) = -10$, *de plus on peut définir la fonction de coût terminale :*

$$g(511) = -40, g(666) = -30, g(611) = -20, g(111) = -10, g(421) = 0.$$

De cette manière, le but du joueur sera de trouver le chemin le moins cher vers les états cibles.
♣

Voici la définition formelle d'une instance du problème SSP.

Définition 3.4. Le problème du plus court chemin stochastique
Une instance du problème du plus court chemin stochastique (SSP) est la donnée d'un uplet $\Pi = (S, A, p, R, c, g)$ *comprenant un processus de décision markovien classique* (s, A, p) *et un ensemble d'états cibles* R, *une fonction de coût des transitions* c *et une fonction de coût terminale* g.

De plus nous dirons qu'une instance du problème SSP est positive si (resp négative) si pour tout $s \in S$ et $a \in A(s)$ on ait $c(s,a) \geq 0$) (resp $c(s,a) \leq 0$).

Le problème consiste donc à trouver le coût minimum pour atteindre R quand on considère uniquement des politiques qui permettent d'atteindre R avec probabilité 1. On peut ainsi définir la notion de *politique adéquate*.

Définition 3.5. Politique adéquate
Soit $T_R(\omega) = min\{k | X_k(\omega) \in R\}$ *le temps de la première visite dans* R. *Pour tout* $s \in S$ *on peut définir l'ensemble des politiques adéquates pour* s *par* $Ad(s) = \{\eta \in Pol | \mathbb{P}_s^\eta(T_R < \infty) = 1\}$.

Il reste maintenant à définir le coût d'une politique par

$$v_s^\eta = E_s^\eta\{g(X_{T_R}) + \sum_{k=0}^{T_R-1} c(X_k, Y_k)\}.$$

Avec toutes ces notations, notons que le problème SSP consiste à :
- trouver l'ensemble des états ayant au moins une politique adéquate
- Calculer le coût minimum $v_s^* = \inf_{\eta \in Ad(s)} v_s^\eta$ d'une stratégie adéquate sur ces états.

Enfin nous dirons qu'une stratégie η est optimale si $v_s^\eta = v_s^*$. Nous verrons par la suite, comment ce problème peut se décliner en des problèmes essentiels pour le model checking des systèmes concurrents.

3.2 Réductions à SSP

Cette partie est consacrée à l'étude des problèmes qui nous intéressent dans la vérification de systèmes distribués probabilistes (ou concurrents), et notamment de la réduction de ces problèmes au problème SSP. Dans ce chapitre nous donnerons essentiellement les résultats théoriques sur ces trois problèmes, nous donnerons des exemples d'applications dans les chapitres suivants.

3.2.1 Problèmes de la probabilité d'accessibilité maximale et minimale

La description de ces problèmes est faite dans la définition 2.41. Rappelons juste qu'il consiste à trouver la probabilité maximale (resp minimale) pour atteindre un ensemble d'états donné (que nous noterons U).
Plus formellement il s'agit de calculer les deux quantités suivantes :

$$u_s^+ = \sup_{\eta \in Pol} \mathbb{P}_s^\eta(reach(U)) \qquad u_s^- = \inf_{\eta \in Pol} \mathbb{P}_s^\eta(reach(U))$$

Posons, (toujours en suivant les notations de [dA99]) $Z \subseteq S$ le sous-ensemble des états ne permettant pas d'atteindre U ($u_s^+ = 0$ sur Z). D'après [CY90], on sait que les problèmes de probabilité d'accessibilité maximale et minimale peuvent se résoudre en utilisant un problème de programmation linéaire sous contraintes sur les variables $\{u_s | s \in S - \{U \cup Z\}\}$. De plus dans [dA97a], l'auteur prouve aussi que les problèmes d'accessibilité admettent des stratégies optimales markoviennes.
Dans cette partie nous allons réduire ces trois problèmes au problème SSP.
Avant de donner la réduction, nous définissons l'opérateur *Reach*.

Définition 3.6. (Opérateur Reach)
Soit E un ensemble d'états. On définit $Reach(E)$ comme l'ensemble des états ayant au moins une politique adéquate pour E. Formellement :

$$Reach(E) = \{s \in S | \exists \eta \in Pol, \mathbb{P}_s^\eta(T_R < \infty) = 1\},$$

Probabilité d'accessibilité maximale

Soit une instance $\Lambda = (S, A, p, U)$ du problème de la probabilité d'accessibilité maximale. On construit une instance du problème SSP par $Ssp^+(\Lambda) = (S, A, p, R, c, g)$, dans laquelle $R = Reach(U) \cup Z$, le coût des transitions c est nul partout, et le coût terminal est tel que $g(s) = -1$ pour $s \in Reach(U)$ et $g(s) = 0$ pour $s \in Z$. On a donc défini ainsi à la fois une instance positive et négative du problème SSP.

Théorème 3.7. *Si $\Pi = Ssp^+(\Lambda)$, alors $u_s^+ = -v_s^*$ pour tout $s \in S - R$ où u_s^+ est calculé sur Λ et v_s^* sur Π.*

Démonstration. Etant donnés les coûts définis nous avons pour une politique η donnée : $v_s^\eta = -E_s^\eta[\mathbf{1}_{X_{T_R} \in U}]$ soit donc $v_s^\eta = -\mathbb{P}_s^\eta[X_{T_R} \in U]$. De ce fait on a bien $u_s^+ = -v_s^*$. en choisissant ces coûts. □

Il reste tout de même à savoir comment calculer l'ensemble $Reach(U)$. Ceci se fait en temps quadratique en la taille du processus de décision markovien et la formule (sous la forme d'une formule de μ-calcul se trouve dans [dA99]).

Exemple 3.8. *Appliquons ce résultat à un exemple encore plus simple que celui du jeu de dés 421, nous décrirons le processus de décision markovien par la représentation graphique de la figure 3.1 et par le tableau des différentes lois de probabilités.*

$t =$	A	B	Bon	$Mauvais$
$p_{A,t}(a)$	0	0	9/10	1/10
$p_{B,t}(b)$	0	1	0	0
$p_{B,t}(c)$	1/2	1/2	0	0
$p_{B,t}(d)$	0	0	1/2	1/2

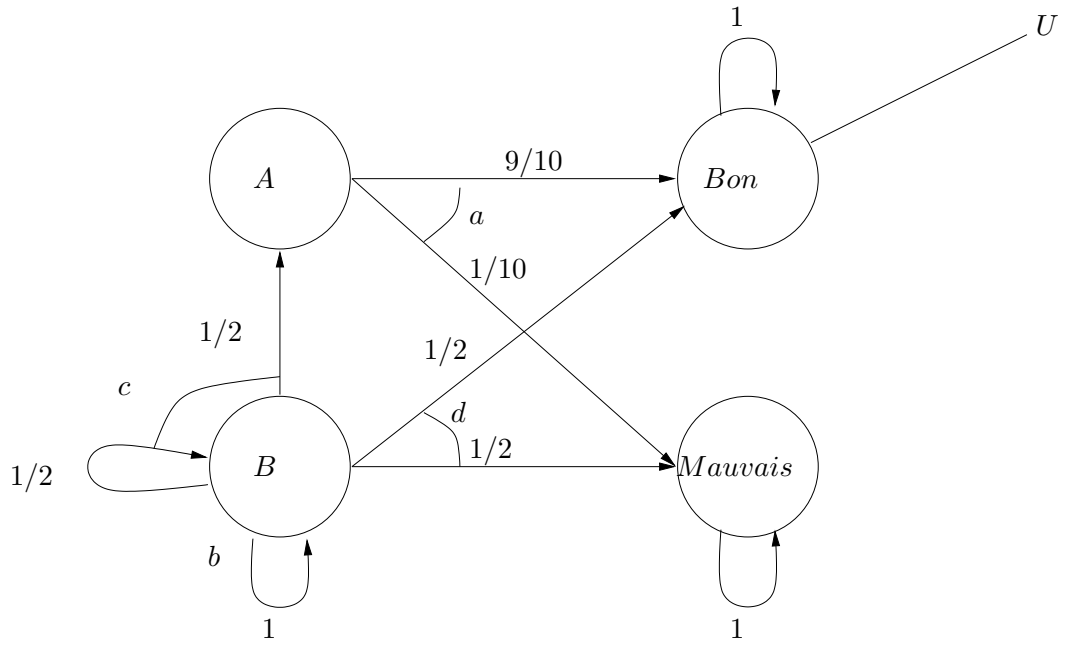


FIG. 3.1 – Un exemple simple

Dans cet exemple très simple, nous noterons $U = Bon$ l'état que l'on cherche à atteindre. Il est évident de constater que $Reach(U) = U$, en effet il n'y a aucune politique adéquate pour U du fait que les politiques a ou d peuvent mener à $Mauvais$ et les politiques c ou d ne mènent pas à $U = Bon$. Ainsi, si on applique le théorème 3.7 à notre exemple pour résoudre problème d'accessibilité maximale à $U = Bon$, on doit résoudre le problème SSP avec $R = Bon \cup Mauvais$ ($Z = Mauvais$ ici). Ainsi, le nouveau processus de décision markovien que l'on doit étudier est donné en figure 3.2.

♣

Probabilité d'accessibilité minimale

De manière analogue, nous pouvons réduire le problème de la probabilité d'accessibilité minimale en un problème SSP à la fois positif et négatif.
Il est nécessaire tout d'abord de trouver l'ensemble des composants finaux maximaux entièrement à l'extérieur de U. Notons ces composants $\{(C_1, D_1), ..., (C_n, D_n)\}$, et notons $C = \cup_{i=1}^{n} C_i$. Il est évident que depuis les éléments de $Z \cup C$ la probabilité minimale d'atteindre U est nulle. Revenons maintenant à la réduction elle-même. Etant donnée une instance $\Lambda = (S, A, p, U)$ du problème de la probabilité d'accessibilité minimale, on construit une instance $Ssp^{-}(\Lambda) = (S, A, p, R, c, g)$ du problème SSP avec $R = U \cup Z \cup C$, le coût des transitions (c) est identiquement nul, et le coût terminal est défini par $g(s) = 0$ pour $s \in Z \cup C$, et $g(s) = 1$ pour $s \in U$. Ainsi, nous avons le théorème suivant :

Théorème 3.9. *Si $\Pi = Ssp^{-}(\Lambda)$, alors $u_s^- = v_s^*$ pour tout $s \in S - R$ où u_s^- est calculé sur Λ et v_s^* sur Π.*

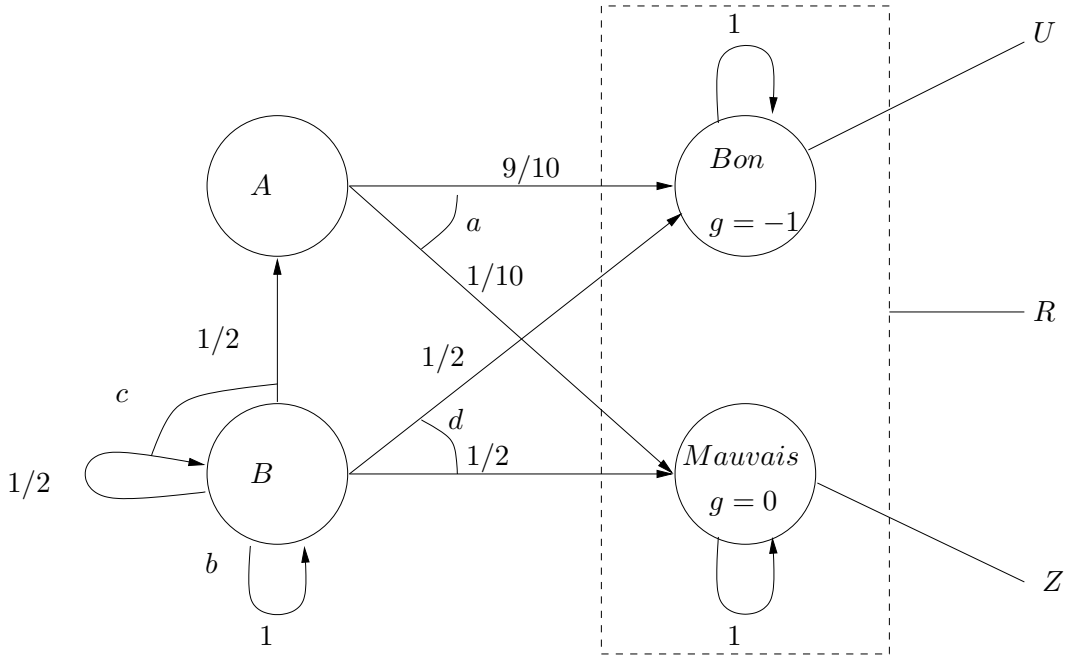


FIG. 3.2 – Problème de l'accessibilité maximale réduit à SSP

Démonstration. De la même manière que précédemment nous calculons v_s^η pour les coûts définis et on obtient bien $v_s^\eta = \mathbb{P}[X_{T_R} \in U]$ et ainsi on a $u_s^- = v_s^*$. Remarquons simplement aussi, que dans ce cas, toutes les stratégies de Π sont optimales. □

Exemple 3.10. *Reprenons notre exemple simple décrit dans la partie précédente. Pour résoudre le problème d'accessibilité minimale nous devons trouver les composants finaux maximaux n'intersectant pas avec $U = Bon$. Il n y en a qu'un : $(B, \{b\})$. Ainsi pour résoudre le problème d'accessibilité minimale, il faut donc s'intéresser à un problème SSP ayant le graphe de la figure 3.3.*

Notons que pour résoudre le problème SSP on peut retirer toutes les transitions qui partent des éléments de R.

♣

3.2.2 Problème du pire temps moyen d'atteinte

La définition de ce problème est donné dans le chapitre précédent dans la définition 2.44. Il s'agit de calculer le pire temps moyen avant d'atteindre un ensemble U. Soit donc $\Lambda = (S, A, p, U)$ une instance du problème du pire temps moyen d'atteinte. On construit une instance négative $Ssp^t(\Lambda) = (S, A, p, U, c, g)$ (notons que $R = U$) du problème SSP avec g identiquement nul et $c(s, a) = -1$ pour tout $s \in S - R$ et $a \in A(s)$. Notons simplement que si on avait voulu résoudre le problème du meilleur temps d'atteinte il aurait fallu prendre $c(s, a) = 1$. On a donc le résultat suivant.

Théorème 3.11. *Si $\Pi = Ssp^t(\Lambda)$, alors $T_s^p = -v_s^*$ pour tout $s \in S - R$ où T_s^p est calculé sur Λ et v_s^* sur Π.*

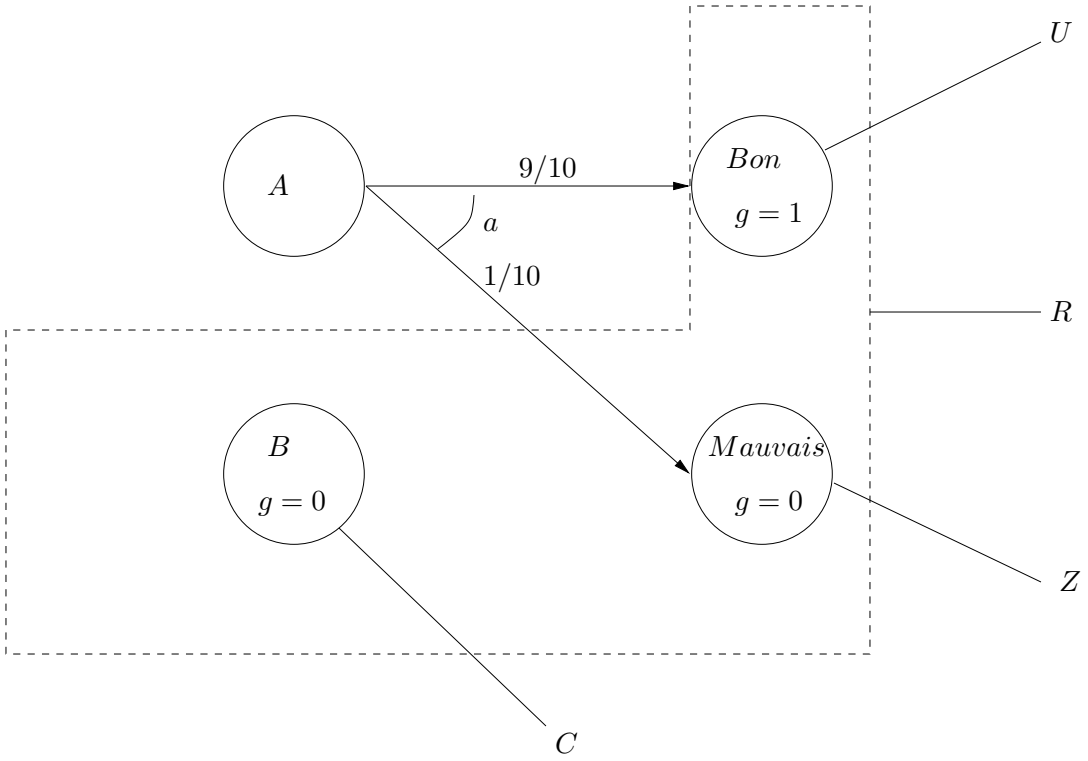


FIG. 3.3 – Problème de l'accessibilité minimale réduit à SSP

Démonstration. La preuve est assez évidente, en effet $u_s^\eta = E_s^\eta(T_U) = E(X_\eta) = T_s^p$. □

Il faut remarquer ici que la solution d'une instance du pire temps moyen appartient à $\mathbb{R} \cup +\infty$, en effet pour que ce temps soit fini, il faudrait que $Reach(U) = S$, c'est à dire que quel que soit l'état de départ, la probabilité d'arriver en U en un temps fini est 1. Ce n'est pas forcément le cas en général et notamment dans le simple exemple que nous donnons pour illustrer les réductions précédentes.

Exemple 3.12. *Pour illustrer le problème du pire temps moyen d'atteinte, nous allons prendre un exemple dans lequel $Reach(U) = S$. Considérons donc le processus de décision markovien dont la représentation graphique est donné dans la figure 3.4*

Pour la réduction du problème du pire temps d'atteinte à SSP, les choses sont beaucoup plus simples, en effet comme $R = U$ il n'y a aucun ensemble à calculer. Le graphe du processus de décision markovien correspondant au problème SSP est donc celui de la figure 3.5.

♣

3.3 Résoudre SSP

Les problèmes auxquels nous nous intéressons sont donc facilement réductibles au problème SSP. Nous allons maintenant présenter les principales manières de résoudre le problème SSP et notamment celle utilisant l'opérateur de Bellman. Les formalismes utilisés dans les parties suivantes sont extraits des deux articles [dA99, BT91].

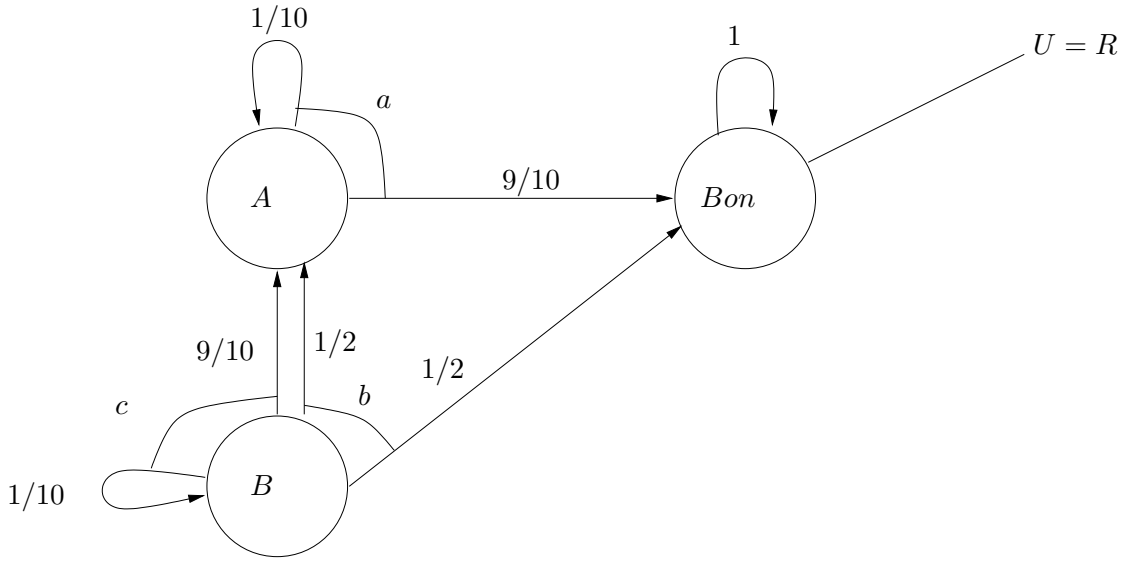


FIG. 3.4 – Un exemple pour le problème du pire temps moyen

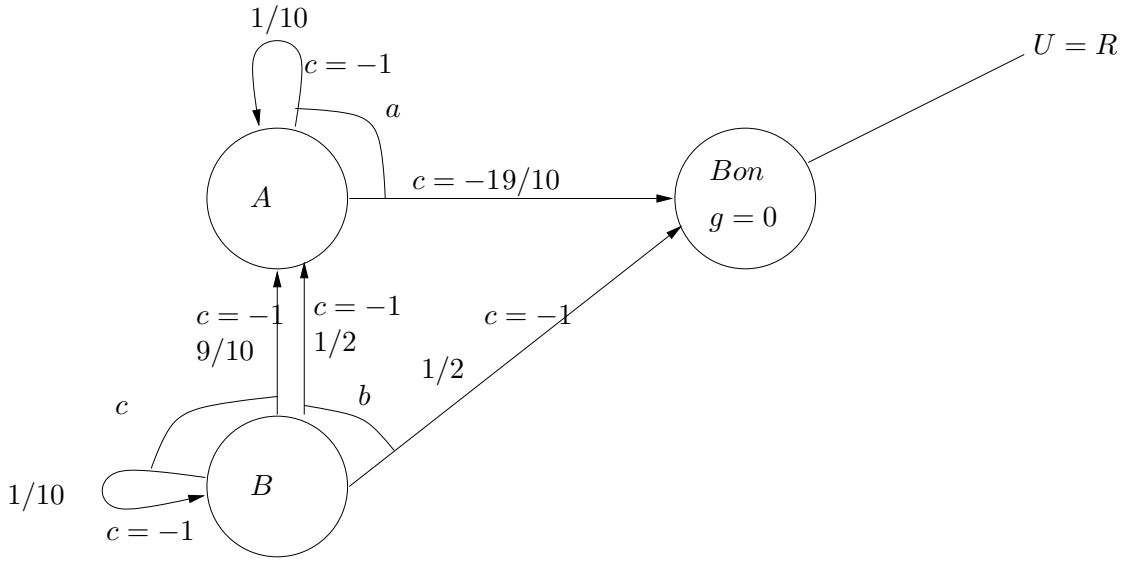


FIG. 3.5 – Le problème du pire temps moyen réduit à SSP

3.3.1 Cas général

Ici, nous présentons une solution au problème SSP suivant les hypothèses et les notations faites dans [BT91, dA97a].
La solution donnée repose sur deux hypothèses fondamentales :

1. **SSP1 :** Il y a toujours une politique adéquate qui est markovienne.
2. **SSP2 :** Si η est une politique markovienne non adéquate alors $v_s^\eta = \infty$ pour au moins un $s \in S - R$.

Les résultats suivants sont tirés de [BT91], dans lequel on peut consulter les preuves.

Théorème 3.13. (Equations de Bellman)
Soit (S, A, p, R, c, g) une instance du problème SSP. Notons $v = [v_s]_{s \in S-R}$ un vecteur de réels et définissons la fonctionnelle L (Opérateur de Bellman) par :

$$[Lv]_s = min_{a \in A(s)}[c(s,a) + \sum_{t \in S-R} p_{st}(a)v_t + \sum_{t \in R} p_{st}(a)g(t)], \tag{3.1}$$

pour tout $s \in S - R$

Si SSP1 et SSP2 sont vérifiées alors toutes les propositions suivantes sont vraies :

- *La fonctionnelle L admet un unique point fixe $v^\diamond$ telle que $v^\diamond = Lv^\diamond$*
- *Le point fixe $v^\diamond$ est la solution unique du problème de programmation linéaire suivant : Maximiser $\sum_{s \in S-R} v_s$ sous les contraintes :*

$$v_s \leq c(s,a) + \sum_{t \in S-R} p_{st}(a)v_t + \sum_{t \in R} p_{st}(a)g(t) \tag{3.2}$$

- *On a $v^* = v^\diamond$ pour tout $s \in S - R$*
- *Si l'on considère une politique markovienne n'autorisant que les actions qui réalise le minimum de l'opérateur de Bellman, soit :*

$$Q_\eta(a,s) > 0 \qquad ssi \qquad a \in arg \min_{a \in A(s)} [c(s,a) + \sum_{t \in S-R} p_{st}(a)v_t + \sum_{t \in R} p_{st}(a)g(t)]$$

pour tout $s \in S - R$. Alors, cette politique η est adéquate, et on a : $v_s^\eta = v_s^* = v_s^\diamond$.
Note : La notation arg représente l'action pour laquelle l'opérateur de Bellman est minimal.

Ce théorème nous permet de déterminer deux manières pour résoudre le problème SSP, soit par itération de l'opérateur de Bellman de l'équation 3.1 (i.e. on calcule $L^n v$) afin de trouver un point fixe, soit par des techniques de programmation linéaire afin de résoudre le problème de programmation linéaire sous contraintes équivalent (équation 3.2).

Application : Problème du pire temps moyen d'atteinte

Il est assez évident de constater que pour les problèmes intéressants de calcul de pire temps moyen d'atteinte l'hypothèse SSP1 est vérifiée car sinon le pire temps serait infini. De plus, étant donnée l'instance du problème SSP obtenu après réduction (Ssp^t), si une politique n'est pas adéquate alors on a facilement $v_s^\eta = +\infty$ (en effet, $c(s,a)$ vaut identiquement 1 ailleurs que dans U) et donc le problème SSP induit par l'instance du problème du pire temps moyen d'atteinte vérifie bien les hypothèses d'application du théorème précédent. Notons que le pire temps moyen d'atteinte est donné par la fonction : $\sup_{s \in S-U}(-v_s^*)$.

Exemple 3.14. *Reprenons l'exemple simple décrit en 3.12 et calculons le pire temps moyen d'atteinte. Tout d'abord calculons le vecteur v_s^*, en trouvant le point fixe de l'opérateur de Bellman. Partons de l'état initial $v_A = v_B = 0$ on obtient aisément une suite pour v_A : $v_A^{n+1} = -1 + \frac{1}{10} v_A^n$, en effet $c(s,a) = -1$, $g = 0$ et il n'y a qu'une seule action possible a. Ainsi $v_A^* = -\frac{10}{9}$ et donc le temps moyen partant de A est de $10/9$. Intéressons nous, à présent au*

cas plus intéressant de B, en prenant comme valeurs initiales $v_A = -\frac{10}{9}$ et $v_B - 0$. Dans ce cas, l'effet de l'opérateur de Bellman ne change pas la valeur de v_A. On a donc :

$$v_B^{n+1} = \min_{i \in \{b,c\}} [-1 + p_{BB}(i)v_B^n + p_{BA}(i)v_A^n]$$

Soit,

$$v_B^{n+1} = \min\{-1 - \frac{10}{9}\frac{1}{2}, -1 - \frac{10}{9}\frac{9}{10} + \frac{1}{10}v_B^n\}.$$

Or, on démontre aisément par récurrence que v_B^n est négatif et donc on en conclut que

$$v_B^{n+1} = -2 + \frac{1}{10}v_B^n.$$

Il a donc pour point fixe $v_B^* = -\frac{20}{9}$. La solution du problème du pire temps moyen pour cet exemple est donc de $\frac{20}{9}$.

♣

Application : Problème de la probabilité d'accessibilité minimale

Il est encore plus simple ici de remarquer que les hypothèses SSP1 et SSP2 sont vérifiées. En effet, pour tout s toutes les politiques sont adéquates.

Exemple 3.15. *Appliquons donc la méthode pour calculer la probabilité minimale d'atteinte de U dans l'exemple donné dans la partie 3.2.1. Bien sûr, pour les états qui sont dans R et pas dans U (soit $Mauvais$ et B) la probabilité minimale d'atteinte est* 0*. Reste donc juste à savoir la probabilité minimale pour l'état A. Il se trouve que dans ce cas le calcul est évident est nous obtenons que la probabilité minimale partant de A est de* 9/10*. (L'opérateur de Bellman dans cet exemple est constant et vaut* 9/10*).* ♣

Malheureusement, les hypothèses SSP1 et SSP2 sont assez contraignantes (notamment fausses dans le cas général pour le problème d'accessibilité maximale). Il est donc nécessaire de s'intéresser à des sous-classes du problème SSP nécessitant des hypothèses moins contraignantes.

3.3.2 Cas particulier : Instances positives

Plaçons nous dans le cas où l'instance du problème SSP est positive (les coûts de transitions sont positifs, les coûts finaux sont quelconques). Dans ce cas particulier, Luca de Alfaro [dA99] nous permet de résoudre facilement le problème SSP. Nous donnons ici les résultats principaux. Remarquons tout d'abord que l'hypothèse SSP1 se doit d'être vérifiée. Par contre il existe bien des cas où l'hypothèse SSP2 n'est pas vérifiée, cela vient du fait qu'il peut y avoir des composants finaux (à intersection vide avec R) de coûts nuls. On pourrait donc trouver une politique non adéquate dont le coût ne tendrait pas vers l'infini. Ce problème est très bien illustré par la figure 3.2, en effet le composant final $(B, \{b\})$ a un coût nul. Le théorème 3.13 ne s'applique donc pas. En fait, l'opérateur de Bellman n'a pas un unique point fixe dans ce cas. La technique de de Alfaro consiste uniquement à éliminer les composants finaux de coûts nuls qui sont en dehors de R. Pour ce faire il applique un algorithme d'élimination (noté $ElimEC$) qui remplace une instance Π du problème SSP en une autre instance $\overset{\wedge}{\Pi} = ElimEC(\Pi)$ qui

vérifiera les hypothèses du théorème. Donnons ici l'algorithme.

Algorithme d'élimination des composants finaux

Entrée : Une instance $\Pi = (S, A, p, R, c, g)$ du problème SSP.

Sortie : L'instance $\overset{\wedge}{\Pi} = (\hat{S}, \hat{A}, \hat{p}, R, \hat{C}, \hat{g})$ modifiée.

L'algorithme : Pour chaque $s \in S - R$, soit $D(s) = \{a \in A(s) | c(s,a) = 0\}$, et soit $\{(B_1, D_1), ..., (B_n, D_n)\}$ l'ensemble des composants finaux maximaux à l'extérieur de R et de coûts nuls. On définit alors $\hat{S} = S \cup \{\hat{S}_1, ..., \hat{S}_n\} - \cup_{i=1}^{n} B_i$. Les ensembles d'actions associés aux nouveaux états sont :

- Si $s \in S - \cup B_i$ alors $\hat{A}(s) = \{< s, a > | a \in A(s)\}$
- Pour $1 \leq i \leq n$ $\hat{A}(\hat{S}_i) = \{< s, a > | s \in B_i \wedge a \in A(s) - D_i(s)\}$

Pour $s \in \hat{S}, t \in S - \cup B_i$ et $< u, a > \in \hat{A}(s)$, les transitions probabilistes sont données par $\hat{p}(s, < u, a >)(t) = p(u,a)(t)$ et $\hat{p}(s, < u, a >)(\hat{S}_i) = \sum_{t \in B_i} p(u,a)(t)$. Enfin, pour $s \in \hat{S}$ et $< u, a > \in \hat{A}(s)$ on définit $\hat{C}(s, < u, a >) = c(u,a)$, g restant inchangé.

Intuitivement, on remplace les composants finaux de coût zéro par des états uniques et on autorise uniquement les transitions qui permettent éventuellement de les quitter.

Donnons à présent les résultats obtenus grâce à cette simplification.

Théorème 3.16. (Résolution des instances positives de SSP) *Soit* Π *une instance positive du problème SSP qui vérifie l'hypothèse SSP1, et soit* $\overset{\wedge}{\Pi} = ElimEC(\Pi)$ *l'instance après avoir avoir éliminé les composants finaux de poids nul, alors on a :*

- $\overset{\wedge}{\Pi}$ *vérifie SSP1 et SSP2*
- $v_s^* = \hat{v}_s^*$ *pour tout* $s \in S - \cup B_i$
- $v_s^* = \hat{v}_{s_i^*}$ *pour tout* $s \in \cup B_i$
- *L'instance* Π *admet une politique optimale markovienne.*
- *La solution du problème SSP est le plus grand point fixe de l'opérateur de Bellman. De plus c'est aussi la solution unique du problème de programmation linéaire sous contraintes suivant :*
  Maximiser $\sum_{s \in S-R} v_s$
  sous la contrainte

$$v_s \leq c(s,a) + \sum_{t \in S-R} p(s,a)(t) v_t + \sum_{t \in R} p(s,a)(t) g(t).$$

Nous ne détaillerons pas la preuve ici, qui est donnée dans [dA97a, dA99].

3.3.3 Cas particulier : Instances négatives

Il existe aussi un résultat analogue pour les instances négatives du problème SSP. la différence avec le cas positive est que l'on peut avoir des coûts valant $-\infty$, la première étape

consiste donc à calculer l'ensemble des états dont le coût minimum pour atteindre R est $-\infty$ puis ensuite d'appliquer l'algorithme d'élimination des composants finaux. Pour nous simplifier la tâche, nous considérerons uniquement le cas où l'ensemble des états ayant un coût optimal $-\infty$ est vide. Sous cette hypothèse nous avons le résultat suivant.

Théorème 3.17. (Résolution des instances positives de SSP) *Soit Π une instance positive du problème SSP qui vérifie l'hypothèse SSP1, et soit $\overset{\wedge}{\Pi} = ElimEC(\Pi)$ l'instance après avoir avoir éliminé les composants finaux de poids nul, alors on a :*

- *$\overset{\wedge}{\Pi}$ vérifie SSP1 et SSP2*
- *$v_s^* = \overset{\wedge}{v_s^*}$ pour tout $s \in S - \cup B_i$*
- *$v_s^* = \hat{v}_{\overset{\wedge}{s_i^*}}$ pour tout $s \in \cup B_i$*
- *L'instance Π admet une politique optimale markovienne.*
- *La solution du problème SSP est le plus grand point fixe de l'opérateur de Bellman, de plus c'est aussi la solution unique du problème de programmation linéaire sous contraintes suivant :*
  Maximiser $\sum_{s\in S-R} v_s$ sous la contrainte $v_s \leq c(s,a)+\sum_{t\in S-R} p(s,a)(t)v_t+\sum_{t\in R} p(s,a)(t)g(t)$.
- *Enfin, la solution du problème SSP est donnée dans ce cas par $\lim_{k\to\infty} L^k(0)$, où 0 est le vecteur nul.*

Application Probabilité d'accessibilité maximale

Le seul problème que nous n'arrivions pas à résoudre directement est celui de l'accessibilité maximale. A présent nous pouvons le faire, en nous servant du théorème de résolution des instances positives de SSP. Faisons le sur un exemple.

Exemple 3.18. *Reprenons le processus de décision markovien décrit par la représentation graphique de la figure 3.1 et tentons de trouver la probabilité maximale d'atteindre Bon, il suffit donc de résoudre le problème SSP décrit en figure 3.2 et pour cela d'éliminer d'abord les composants finaux, soit le nouveau graphe suivant 3.6.*

Nous pouvons donc à présent calculer la solution de ce problème SSP v_s^*. Il est évident de calculer v_A^*, car il n y a pas le choix entre plusieurs actions, ainsi $v_A^* = -\frac{9}{10}$ et donc la probabilité maximale partant de A pour atteindre Bon est évidemment de $\frac{9}{10}$.
Partant de B on a :

$$v_B^{n+1} = \min\{\frac{1}{2}v_B^n - \frac{1}{2}\frac{9}{10}, -\frac{1}{2}\}$$

On peut aisément montrer par récurrence que $v_B^n \leq \frac{1}{2}$, ainsi

$$v_B^{n+1} = \frac{1}{2}v_B^n - \frac{1}{2}\frac{9}{10}$$

Et donc

$$v_B^* = -\frac{9}{10}$$

Soit la probabilité maximale d'atteindre Bon partant de B est de $\frac{9}{10}$ ♣

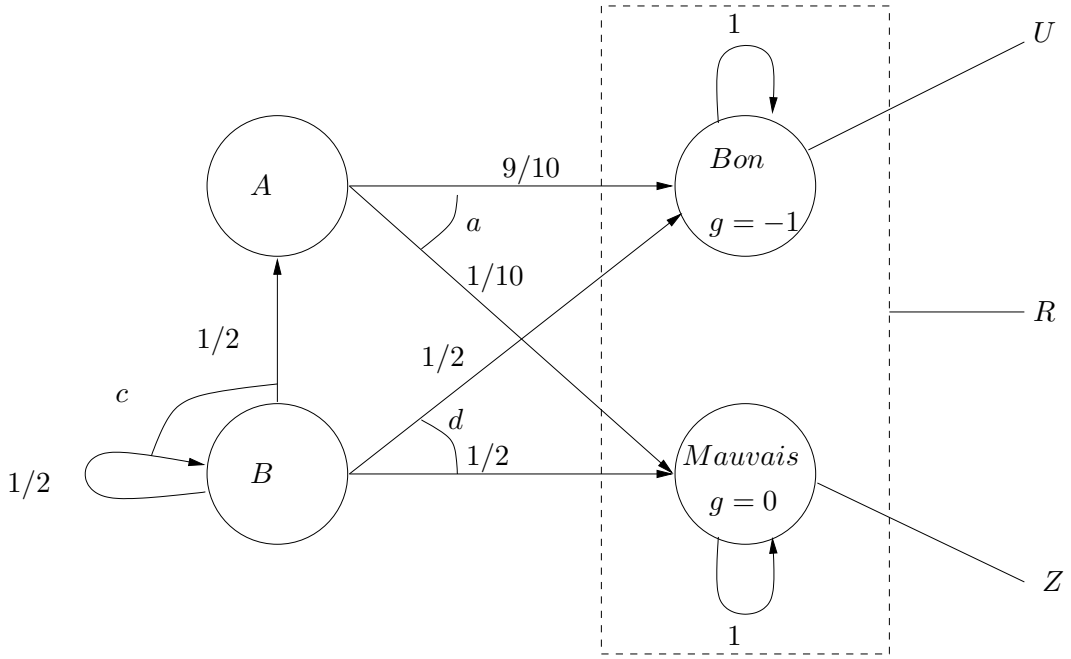


FIG. 3.6 – Le problème de la probabilité maximale réduit à SSP

3.3.4 Détermination de la politique optimale

La connaissance de la politique optimale est souvent un enjeu majeur. Dans des problèmes de théorie des jeux ou d'économie, c'est souvent la question principale. Les théorèmes que nous avons décrits donnent un résultat important sur cette politique, en effet on sait qu'elle est markovienne, ce qui n'est pas négligeable. De plus, Bertsekas et Tsitsiklis nous donnent deux heuristiques qui permettent de calculer (ou d'approcher) cette politique [BT91].

Méthode par approximation successives

Nous partons d'un vecteur v d'éléments de $S - R$, on désire trouver la politique optimale, pour une instance donnée du problème SSP. Tout d'abord définissons

$$L_\mu v = c(s,a) + \sum_{t \in S-R} p_{st}(a) v_t + \sum_{t \in R} p_{st}(a) g(t)$$

nous allons calculer la suite μ^k telle que x $L_{\mu^k}[L^{k-1}v] = L^k v$ pour tout k. Il suffit donc de trouver à chaque étape la politique qui minimise l'opérateur de Bellman. Or, d'après le théorème 3.13, $\lim_{k\to\infty} L^k v = v^*$. On obtient une politique μ optimale en passant à la limite.

Méthode par itération

Dans cette méthode nous partons d'une politique markovienne quelconque μ et du coût correspondant v^μ et nous calculons μ' tel que $L_{\mu'} v^\mu = L v^\mu$, soit de manière équivalente, pour tout $s \in S - R$:

$$\mu'(s) = arg \min_{a \in A(s)} [c(s,a) + \sum_{t \in S-R} p_{st}(a) v_t^\mu + \sum_{t \in R} p_{st}(a) g(t)].$$

On remarque que la nouvelle politique est strictement meilleure que l'ancienne, si tel n'était pas le cas, on aurait déjà atteint la politique optimale, et donc cet algorithme itératif nous donne une politique optimale en un nombre fini d'étapes car $A(s)$ est fini.

3.3.5 Fondements du model-checker : PRISM

L'ensemble de ces résultats forment le fondement théorique du model checking probabiliste. Ces fondements se trouvent par exemple dans [BdA95, Bai98, BK96]. Dans la partie suivante, nous allons utiliser le model-checker PRISM ([KNP02]) pour étudier des processus de décision markoviens. Il est donc nécessaire de décrire les algorithmes principaux sur lesquels le model checker est fondé. Nous allons décrire ici comment PRISM fonctionne pour résoudre les problèmes simples auxquels nous sommes confrontés.
Etant donné un ensemble fini $F \subset S$ d'états cibles, on notera, dans cette partie, $P_\mu(s \rightarrow^* F)$ la probabilité d'atteindre F partant de s sous la politique μ.
Nous nous intéresserons plus particulièrement aux problèmes de la probabilité d'accessibilité maximale (resp minimale), qui sont, rappelons le, définies par :

$$P_{max}(s \rightarrow^* F) = \max_\mu \{P_\mu(s \rightarrow^* F)\}$$

$$P_{min}(s \rightarrow^* F) = \min_\mu \{P_\mu(s \rightarrow^* F)\}$$

On pourrait vérifier d'autres propriétés sur les processus de décision markoviens, en effet dans [BdA95] les auteurs étendent les algorithmes classiques de model checking pour toutes les propriétés de PCTL (extension probabiliste de la logique classique CTL). En ce qui nous concerne, nous nous intéresserons uniquement au problème de la probabilité maximale d'accessibilité (resp. minimale). Ces problèmes, comme nous l'avons vu dans ce chapitre, se réduisent aisément à un problème SSP, et PRISM se doit donc uniquement de résoudre le problème SSP correspondant.
Le choix fait par les concepteurs de ce model-checker est de résoudre le problème de programmation linéaire 3.2 rappelé dans [BdA95]. En effet, cela semble plus systématique, puisqu'il n'est pas nécessaire d'éliminer d'abord les composants finaux de poids nul. Ceci peut se faire en temps polynomial sur la taille du processus de décision markovien.
Pour résoudre ce problème classique de programmation linéaire sous contraintes, il existe deux options, soit la solution d'un problème d'optimisation linéaire en utilisant par exemple la méthode connue du simplexe [KNSS02], soit par une méthode d'approximation successive par itération. C'est la deuxième option que les concepteurs de PRISM ont choisi.
Pour analyser un processus de décision markovien, PRISM construit complètement le graphe des états accessibles, et la matrice de transition que cela représente. Dans un souci d'éviter le phénomène d'explosion combinatoire, il utilise des techniques de stockage de données de type BDD (diagrammes de décision binaires) appelées MTBDD (multi terminal BDD). Ceci permet de compacter la taille du système, et d'exploiter ses propriétés haut-niveau.
Enfin la toute dernière version de PRISM (encore un prototype) permet l'analyse de quantité basé sur des coûts. Par exemple, dans le problème du pire (ou du meilleur) temps moyen d'atteinte, on calcule la valeur moyenne d'un coût. Utilisant les algorithmes de model-checking décrit dans [dA97a] (basé aussi sur la résolution du problème SSP), les concepteurs du model-checker nous permettent à présent de calculer le coût moyen minimum et le coût moyen maximum que nous noterons par la suite $R_{min}[vraiUF]$ et $R_{max}[vraiUF]$.

Plusieurs exemples d'études de cas sont disponibles sur la page web du model checker PRISM [PRI].

Deuxième partie

Analyse d'accessibilité dans les systèmes temporisés probabilistes

Chapitre 4

Automates temporisés probabilistes

Mais pourquoi courent-ils si vite ?
Pour gagner du temps ! Parce que le temps c'est de
l'argent... Plus ils courent plus ils en gagnent !
Raymond Devos

La présence de probabilités dans un grand nombre de protocoles informatiques (par exemple ceux de communications) est essentielle, parfois même déterminante pour leur bon fonctionnement. Nous pouvons citer plusieurs exemples de ces protocoles ; le protocole IEEE 1394 FireWire étudié dans [KNSS02], le protocole CSMA/CA IEEE 802.11 étudié dans [KNS02], ou le protocole IEEE 802.3 CSMA/CD étudié dans [FMP04d, DFH+04] et dont nous détaillerons les résultats dans le chapitre 5 en font partie. La présence de probabilités dans ces protocoles temporisés est souvent nécessaire, comme dans la plupart des cas ou les probabilités sont introduites, pour briser des symétries et pour assurer certaines propriétés de convergence Les modélisations classiques (automates temporisés, chaînes de Markov, processus de décision markoviens) ne permettent pas de les décrire convenablement. Il est donc nécessaire d'introduire une nouvelle notion qui combinera à la fois la présence de temps (symbolisés par des horloges), de choix probabiliste et de non déterminisme : les automates probabilistes temporisés (ATP). C'est ce formalisme que nous décrirons dans ce chapitre ainsi que la manière d'en étudier ses propriétés. Un grand nombre d'auteurs différents ont défini ces modèles (citons par exemple [ACD91, dA97a, BK96]), mais nous avons choisi ici de suivre les notations de [KNS].

4.1 Définition des automates temporisés probabilistes

Les automates temporisés probabilistes sont une extension naturelle des automates temporisés (voir [AD94]). Un automate temporisé classique consiste en un graphe de contrôle fini, dont les noeuds sont appelés états, et qui est équipé d'un ensemble fini de variables réelles appelés horloges, dont la valeur traduit l'évolution du temps-réel. Les arêtes (appelées transitions) du graphe de contrôle sont soit toujours disponibles ou soit disponibles que sous certaines contraintes sur les horloges. De plus, un ensemble d'horloges peut être remis à zéro, quand une transition est effectuée. Les automates temporisés probabilistes, sont des automates temporisés pour lesquels il existe des distributions de probabilités sur les différentes transitions. Donnons en dans ce paragraphe les définitions formelles.

4.1.1 Syntaxe des automates temporisés probabilistes

Définition 4.1. (Temps et Horloges) .

- *Soit $\mathbb{T} \in \{\mathbb{R}^+, \mathbb{N}\}$ le domaine sur lequel évolue le temps, donc soit les entiers, soit les réels positifs.*

- *Soit $\mathcal{X}$ un ensemble fini de variables appelés horloges qui prennent leurs valeurs sur le domaine $\mathbb{T}$.*

Un élément $v \in \mathbb{T}^{|\mathcal{X}|}$ est appelé une valuation d'horloges. Nous noterons $0 \in \mathbb{T}^{|\mathcal{X}|}$ la valuation d'horloges qui associe 0 à toutes les horloges de $\mathcal{X}$.
Soit $v \in \mathbb{T}^{|\mathcal{X}|}$ une valuation d'horloges et $t \in \mathbb{T}$ une durée, alors la valuation d'horloges $v \bigoplus t$ représente l'incrément du temps (Voir la partie 4.1.2 pour la sémantique).
Enfin nous noterons $v[X := 0]$ la valuation d'horloges obtenue en réinitialisant toutes les horloges de X à zéro et en laissant les autres inchangées.

Définition 4.2. (Zones) *Nous noterons $Zones(\S)$ l'ensemble des zones sur $\mathcal{X}$, qui sont définies par la conjonction de contraintes atomiques de la forme $x \bowtie c$ et $x - y \bowtie c$, pour x, y des horloges $\bowtie \in \{<, >, \leq, \geq\}$, et $c \in \mathbb{N}$. Une zone ξ est dite sans garde diagonale si elle ne comporte pas de contraintes de la forme $x - y \bowtie c$, elle est dite fermée si elle ne comporte pas de contraintes de la forme $x \bowtie c$ et $x - y \bowtie c$ avec $\bowtie \in \{<, >\}$. La valuation d'horloges v satisfait la zone ξ, ce que l'on note $v \lhd \xi$, si et seulement si ξ est vrai après avoir remplacé chaque horloge par sa valeur correspondante.*

Nous pouvons à présent définir les automates temporisés probabilistes

Définition 4.3. (Automates temporisés probabilistes) *Un automate temporisé probabiliste est un 5-uplet $ATP = (L, \mathcal{X}, \Sigma, inv, prob)$ où :*

- *L est un ensemble fini d'états ;*
- *Σ est un ensemble fini d'événements ;*
- *la fonction $inv : L \to Zones(\mathcal{X})$ représentes les invariants ;*
- *l'ensemble fini $prob \subseteq L \times Zones(\mathcal{X}) \times \Sigma \times Dist(2^{\mathcal{X}} \times L)$ représente l'ensemble des relations de transitions probabilistes. Une transition probabiliste prend la forme d'un quadruplet $(l, g, \sigma, p) \in prob$, dans lequel l est l'état de départ de la transition, g est la condition d'activation (ou garde), σ est l'événement. et p est la distribution sur les transitions.*

Enfin, un automate temporisé probabiliste est dit sans garde diagonale (fermé) si toutes les zones utilisées dans sa description sont sans garde diagonale (fermée).

Une configuration d'un automate temporisé probabiliste est défini par un couple (l, v), oú $l \in L$ et $v \in \mathbb{T}^{\mathcal{X}}$ tel que $v \lhd inv(l)$. Si la configuration courante est (l, v), il y a un choix non déterministe entre laisser le temps passer tant que continûment la condition $inv(l)$ est vérifiée, ou faire une transition discrète conformément à une distribution de probabilité dans $prob$ qui a pour état de départ l et dont la condition d'activation g est satisfaite, par la valuation courante des horloges v. Si la transition probabiliste (l, g, σ, p) est choisie, alors la probabilité pour aller à la l'état l' en remettant à zéro les horloges de l'ensemble X est $p(X, l')$. La sémantique des automates temporisés probabilistes, sera plus grandement détaillée dans la partie suivante 4.1.2.

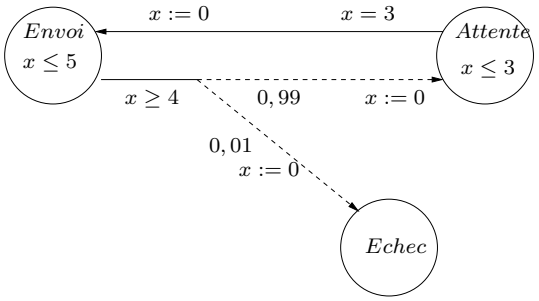


FIG. 4.1 – Un automate temporisé probabiliste simple

La convention graphique des automates temporisés probabilistes est proche de celle des automates temporisés classiques.
Les états sont représentés par des cercles, et les transitions par des flèches. Comme nous le voyons dans la figure 4.1, les gardes, les mises à jours et les probabilités étiquettent la transition alors que les invariants sont donné à l'intérieur des états (dans les cercles). Enfin les transitions probabilistes sont représentés par des flèches en pointillés.

Exemple 4.4. *Voici la description graphique d'un exemple très simple d'automate temporisé probabiliste. Cette exemple est principalement tiré de [DKN02]*

Cet automate représente le comportement d'un émetteur qui essaie d'envoyer un message entre 4 *et* 5 *unité de temps, et si l'envoi et réussi (avec probabilité* $0,99$*) attend trois unités de temps avant de réessayer d'envoyer un nouveau message.*

Nous pouvons dans le cadre de cet exemple donner les caractéristiques de cet automate temporisé probabiliste avec nos notations (voir définition 4.3) :
- $L = \{Envoi, Attente, Echec\}$
- $\mathcal{X} = \{x\}$
- $\Sigma = \emptyset$
- $inv(Envoi) = \{x \leq 5\}$
- $inv(Attente) = \{x \leq 3\}$
- $prob = \{(Envoi, x \geq 4, p)\}$ *dans lequel* $p(\{x\}, Attente) = 0,99$ *et* $p(\{x\}, Echec) = 0,01$

♣

Pour faciliter la modélisation des systèmes complexes que nous allons étudier, il reste à définir les notions d'états urgents et de composition parallèle d'automate temporisé probabiliste.

Définition 4.5. (Etats urgents)
Un état d'un automate temporisé probabiliste est dit urgent, si une fois qu'on y entre on y sort immédiatement, sans laisser le temps passer. La notion d'urgence pour les états est très proche de la notion de transitions urgentes (voir [HHWT97, DOTY96]).
La notion d'états urgents peut facilement s'exprimer en utilisant la définition 4.3, grâce à une horloge supplémentaire des remises à zéros et des conditions d'invariance.

L'illustration de ce concept sera donnée dans le chapitre suivant, lorsque nous étudierons le protocole CSMA/CD.
Il est parfois essentiel, à la fois par un souci de clarté, et aussi du fait de la description des systèmes utilisés de décrire l'automate temporisé probabiliste comme un produit de plusieurs automates temporisés probabilistes, c'est ce que nous appelons la composition parallèle.

Définition 4.6. (Composition parallèle) *La composition parallèle de deux automates temporisés probabilistes PTA_1 et PTA_2 , dans lesquels $\mathcal{X}_\infty \cap \mathcal{X}_\in = \emptyset$, est l'automate temporisé probabiliste $PTA_1 \parallel PTA_2 = (L_1 \times L_2, \mathcal{X}_\infty \cup \mathcal{X}_\in, \pm_\infty \cup \pm_\in, \rangle \backslash \sqsubseteq, \bigvee \nabla \wr \lfloor)$ où $inv(l,l') = inv_1(l) \wedge inv_2(l')$ pour tout $(l,l') \in L_1 \times L_2$ et $((l_1,l_2),g,\sigma,p) \in prob$ si et seulement si une des propositions suivantes est vérifiée :*

- *$\sigma \in \Sigma_1 - \Sigma_2$ et il existe $(l_1,g,\sigma,p_1) \in prob_1$ tel que $p = p_1 \otimes \mu_{(\emptyset,l_2)}$;*
- *$\sigma \in \Sigma_2 - \Sigma_1$ et il existe $(l_2,g,\sigma,p_2) \in prob_2$ tel que $p = \mu_{(\emptyset,l_1)} \otimes p_2$;*
- *$\sigma \in \Sigma_1 \cap \Sigma_2$ et il existe $(l_1,g,\sigma,p_1) \in prob_1$, et $(l_2,g,\sigma,p_2) \in prob_2$ tel que $g = g_1 \wedge g_2$ et $p = p_1 \otimes p_2$,*

où, pour tout $l_1 \in L_1$, $l_2 \in L_2$, $X_1 \subseteq \mathcal{X}_1$ et $X_2 \subseteq \mathcal{X}_2$, on pose $p_1 \otimes p_2(X_1 \cup X_2,(l_1,l_2)) = p_1(X_1,l_1).p_2(X_2,l_2)$.

4.1.2 Sémantique des automates temporisés probabilistes

La sémantique des automates temporisés probabilistes se définit en termes de processus de décision markoviens, la seule différence avec les définitions et les résultats donnés dans le chapitre 2 est que le nombre d'états et de transitions peuvent être infinis. Pour éviter les confusions (les résultats sur les processus de décision markoviens du chapitre 2 étant globalement faux dans le cas infini) nous définissons ici, une notion essentielle équivalente à celle de processus de décision markovien ; les systèmes probabilistes (voir [KNS]). Notons, de plus que cette notion est équivalente à celle d'automates probabilistes de [SL95] et de systèmes probabilistes et non déterministes de [BdA95].

Définition 4.7. (Systèmes probabilistes) *Un système probabiliste $SP = (S, Act, Steps)$ consiste en un ensemble S d'états, un ensemble Act d'actions et une relation de transition probabiliste $Steps \subseteq S \times Acts \times Dist(S)$.*

On effectue une transition probabiliste $s \xrightarrow{a,\mu} s'$ depuis le site $s \in S$ en choisissant d'abord de manière non déterministe un couple action-distribution (a,μ) tel que $(s,a,\mu) \in Steps$, et en faisant ensuite un choix probabiliste de l'état s' conformément à la distribution μ.

Correspondance entre les processus de décision markoviens et les systèmes probabilistes : Soit $SP = (S, Act, Steps)$ on définit le processus de décision markovien correspondant (S', A, p) par :

- $S' = S$,
- Pour tout $s \in S$ on a, $A(s) = \{a \in Act | \exists \mu \in Dist(S), (s,a,\mu) \in Steps\}$,
- Pour tout $s \in S$ et tout $a \in A(s)$ on a $p_{s,s'}(a) = \mu(s')$ avec $(s,a,\mu) \in Steps$

On pourrait de la même manière transformer les processus de décision markoviens en systèmes probabilistes.

Nous pouvons, à présent donner la sémantique des automates temporisés probabilistes en termes de systèmes probabilistes. Comme pour les automates temporisés classiques, il existe

deux sortes de transitions, les transitions temporelles dues à l'écoulement du temps et les transitions discrètes qui correspondent aux transitions probabilistes. La définition suivante est paramétrée à la fois par le domaine temporel $\mathbb{T}$ et par la manière dont le temps s'incrémente $\oplus$.

Définition 4.8. (Sémantique des automates temporisés probabilistes) *La sémantique d'un automate temporisé probabiliste $ATP = (L, \mathcal{X}, \Sigma, inv, prob)$ ayant pour domaine temporel $\mathbb{T}$ et pour incrément temporel $\oplus$ est le système probabiliste $SP^{\oplus}_{\mathbb{T}} = (S, Act, Steps)$ tel que :*

- *$S \subseteq L \times \mathbb{T}^{|\mathcal{X}|}$ tel que $(l, v) \in S$ si et seulement si $v \lhd inv(l)$;*
- *$Act = \mathbb{T} \cup \Sigma$*
- *Steps est l'ensemble de transitions probabilistes définies comme suit, pour tout $(l, v) \in S$:*
  - *Transitions temporelles : Pour chaque durée $t \in \mathbb{T}$, soit $((l, v), t, \mu) \in Steps$ si et seulement si $v \oplus t' \lhd inv(l)$ pour tout $0 \leq t' \leq t$ et $\mu(l, v \oplus t) = 1$;*
  - *Transitions discrètes : Pour toutes les transitions probabilistes $(l, g, \sigma, p) \in prob$, on pose $((l, v), \sigma, \mu) \in Steps$ si et seulement si $v \lhd g$, de plus pour tout $(l'v') \in S$ on a :*

$$\mu(l', v') = \sum_{X \subseteq \mathcal{X} \wedge v' = v[X:=0]} p(X, l'),$$

et pour tout $(X, l') \in 2^{\mathcal{X}} \times L$ tel que $p(X, l') > 0$ on a $v[X := 0] \lhd inv(l')$

Remarques

- Il y a deux classes distinctes de sémantiques qui dépendent de la description du temps qu'on choisit. Si $\mathbb{T} = \mathbb{R}$ alors on note $\oplus = +$ et la sémantique associée est appelée sémantique continue. Par contre si $\mathbb{T} = \mathbb{N}$, on notera $\oplus = \oplus_{\mathbb{N}}$ et la sémantique associée sera appelée sémantique entière. La définition de l'opérateur $\oplus_{\mathbb{N}}$ est comme suit. Soit ATP l'automate temporisé probabiliste ; pour tout $x \in \mathcal{X}$, on notera k_x la plus grande constante à laquelle l'horloge x est comparée dans les zones de ATP. Ainsi pour chaque valuation $v \in \mathbb{N}^{|\mathcal{X}|}$ et pour chaque durée $t \in \mathbb{N}$, on posera $v \oplus_{\mathbb{N}} t$ la valuation d'horloge de $\mathcal{X}$ qui assigne la valeur $\min\{v_x + t, k_x + 1\}$ à toutes les horloges $x \in \mathcal{X}$. La définition de la sémantique entière pour les automates temporisés probabilistes est l'extension naturelle de celle de la sémantique entière pour les automates temporisés classiques voir [Bey01, HMP92, AMT98, BMT99]. Nous omettrons par la suite les distinctions entre $+$ et $\oplus_{\mathbb{N}}$, car une fois le domaine temporel choisi, nous fixerons aussi l'incrémentation, il n'y aura donc pas de confusion.
- On peut noter aussi que la sémantique entière d'un automate temporisé probabiliste est finie, alors que la sémantique continue est généralement infinie. Ces résultats découlant directement des définitions.
- Il n'est pas difficile de vérifier que la sémantique de la composition parallèle de deux automates temporisés probabilistes correspond à la composition parallèle de leur sémantique individuelle. Soit plus formellement $SP1^{\oplus}_{\mathbb{T}} || SP2^{\oplus}_{\mathbb{T}}$ est isomorphe à $S(P1||P2)^{\oplus}_{\mathbb{T}}$ 'a la fois pour la sémantique entière et continue.
- Enfin, nous pouvons remarquer qu'on peut utiliser le même vocabulaire que celui des processus de décision markoviens pour les automates temporisés probabilistes en terme de chemins, de politique (ou adversaire) et de mesure de probabilité. On peut aussi bien sûr parler de probabilité maximale et minimale d'atteinte d'ensembles donnés. Pour plus de détail on peut se reporter au chapitre 2 ou à [KNS].

Enfin terminons cette partie en donnant un théorème fondamental qui découle de résultats similaires établis dans [Bey01], et qui permet d'établir la correction de la sémantique entière. Ce résultat n'est valide que pour les automates temporisés probabilistes fermé et sans gardes diagonales. Nous noterons L' un ensemble d'états cibles de l'automate temporisé probabiliste ATP, et nous noterons $F_{\mathbb{T}}^{L'} = \{(l,v) | l \in L', v \in \mathbb{T}^{|\mathcal{X}|} \wedge v \triangleleft inv(l)\}$ l'ensemble de toutes les configurations correspondantes aux états de L'. Avec ces notations nous avons le théorème suivant (tiré de [KNSS02]).

Théorème 4.9. (Equivalence sémantique entière et continue) *Pour tout automate temporisé probabiliste $ATP = (L, \mathcal{X}, \Sigma, inv, prob)$ fermé et sans garde diagonale, où un état initial l_i et un ensemble d'états cibles L' on a :*

$$MaxProbReach_{SP_{\mathbb{R}}}((l_i,0), F_{\mathbb{R}}^{L'}) = MaxProbReach_{SP_{\mathbb{N}}}((l_i,0), F_{\mathbb{N}}^{L'})$$

$$MinProbReach_{SP_{\mathbb{R}}}((l_i,0), F_{\mathbb{R}}^{L'}) = MinProbReach_{SP_{\mathbb{N}}}((l_i,0), F_{\mathbb{N}}^{L'})$$

avec $MaxProbReach$ et $MinProbReach$ sont les solutions aux problèmes des probabilités d'accessibilité maximale et minimale décrit dans le chapitre 3.

Exemple 4.10. *Reprenons notre exemple simple. Dans ce cas nous pouvons facilement donner les caractéristiques du système probabiliste engendré. Nous prendrons ici le cas où le domaine temporel est discret ($\mathbb{N}$). Nous avons ainsi en suivant les notations de la définition 4.8 :*

- $S = \{(Envoi,0), (Envoi,1), (Envoi,2), (Envoi,3), (Envoi,4), (Envoi,5), (Attente,0), (Attente,1), (Attente,2), (Attente,3), (Echec,0)\}$
- $Act = \mathbb{N}$
- $Steps = \{((Envoi,4),4,\mu), ((Envoi,5),5,\mu)\}$ *dans lequel* $\mu(Erreur,0) = 0,1$ *et* $\mu(Attente,0) = 0,99$. *tous les autres étant nuls*

♣

4.2 Vérification et analyse des automates temporisés probabilistes

La première étape avant l'analyse est le choix du domaine temporel, de manière générale est un temps dense (par exemple $\mathbb{R}$). Malheureusement, dans ce cas l'ensemble des valeurs possibles prises par les horloges est infini. La vérification des modèles infinis étant complexe, il existe un grand nombre de cas, pour lesquels un modèle fini équivalent peut être construit. Ce sont à ces modèles que nous nous intéressons ici. La méthode la plus classique consiste en la détermination du graphe des régions ; une riche littérature traite de ce sujet dans le cas des automates temporisés classiques [AD94], nous examinerons comment cette méthode s'applique aux automates temporisés probabilistes. La seconde consiste en la vérification des propriétés temporelles par exploration en avant, en itérant l'opérateur de succession depuis les états de départ [DOTY96]. Enfin, une autre solution, est de ne pas prendre un domaine de temps dense mais un domaine de temps discret ($\mathbb{N}$), et d'utiliser la sémantique entière, sous la condition que l'automate est fermé et sans garde diagonale.

4.2.1 Remarques sur la logique

Nous avons déterminé au chapitre 3 pourquoi nous nous intéressons au problèmes de la probabilité d'accessibilité maximale et minimale. Ces deux quantités s'expriment facilement dans la logique probabiliste temporelle PCTL [HJ94, BdA95]. Cette logique est obtenue depuis la logique temporelle classique CTL en remplaçant les quantificateurs existentiel et universel $\exists$ et $\forall$, par un quantificateur probabiliste $\mathbb{P}$.
Par exemple, la formule PCTL $\mathbb{P}_{\geq\lambda}(\diamond\Phi)$ est vraie dans un état du système probabiliste si toutes les politiques assignent une probabilité au moins égale à λ aux chemins qui accèdent à un état dans le quel la formule Φ est vérifiée. De cette manière on peut facilement exprimer les propriétés d'accessibilité minimales et maximales.

Nous pouvons à présent décrire les différents modèles abstraits permettant de vérifier ces propriétés.

4.2.2 Analyse en avant

La description de cette méthode est donnée en détail dans [KNS] La méthode par exploration en avant pour la vérification des automates temporisés probabilistes utilise une recherche sur le graphe à travers les états du modèle en utilisant les arêtes, les zones, et les remises à zéro pour déterminer l'ensemble des états accessibles pour la sémantique continue.
Un ensemble de configurations calculé à n'importe quelle étape de l'algorithme est un couple comprenant un état et une zone qu'on appellera état symbolique (suivant la notation de [KNS]). Ainsi pour tout automate temporisé classique ou probabiliste le nombre d'états symboliques est fini, on pourra donc vérifier les propriétés sur ce nouveau modèle. En effet les propriétés temporisés probabilistes d'accessibilité, telle que, par exemple, avec probabilité $0,9$ ou plus, le système atteint un état acceptant en moins de 100 microsecondes, seront vérifiées sur le système probabiliste d'états symboliques obtenu en prenant pour états les états symboliques et pour transitions les transitions dérivés de l'automate temporisé probabiliste [KNSS02].
Nous allons reprendre notre exemple simple pour décrire le système probabiliste d'états symboliques obtenu par analyse en avant.

Exemple 4.11. *Reprenons, de nouveau l'exemple simple dont l'automate est donné en figure 4.1. Nous donnons ici la méthode complète qui permet d'obtenir le processus de décision markovien après accessibilité en avant. Rappelons que nous travaillons sur des états symboliques (Couples état-zone (l, ξ)).*

1. *La première étape est de déterminer les successeurs par transition des états symboliques : Le successeur par transition d'un état symbolique (l, ξ) par rapport à la transition $e = (l, g, X, l')$ (où X représente l'ensemble des horloges remises à zéro lors de la transition de l vers l', la valeur de la probabilité ici n'est pas importante) est :*

$$succ.tran((l.\xi), e) = (l', (\xi \wedge g)[X := 0] \wedge inv(l'))$$

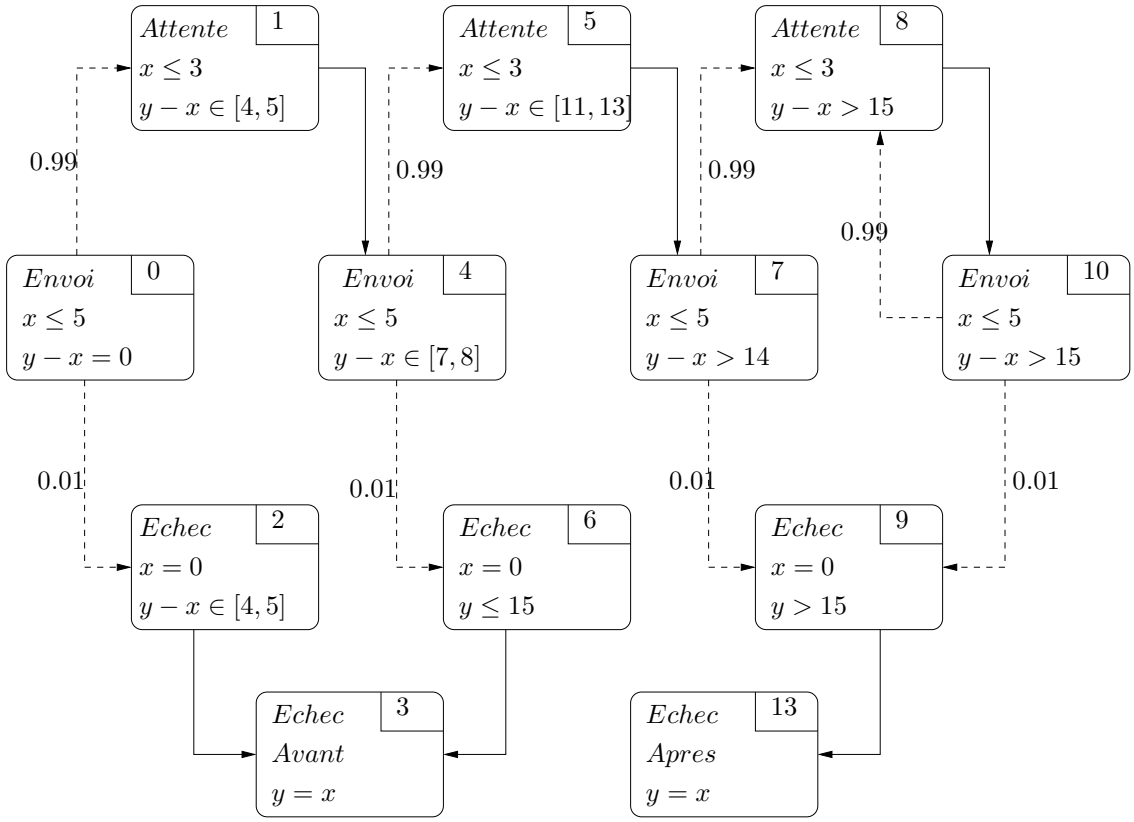


FIG. 4.2 – Graphe obtenu par analyse en avant

2. *Il faut ensuite déterminer les successeurs temporels, ce sont les états symboliques définis par*

$$succ.temp((l, \xi) = (l, \vec{\xi} \wedge inv(l))$$

3. *Ensuite, et afin de rendre le graphe obtenu fini, on se fixe un délai (dans notre exemple* 15 *unités de temps), et on utilise une nouvelle horloge qui permettra de mesurer le temps écoulé. Tous les états pour lesquels cette deuxième horloge (y) dépassera la deadline seront rassemblés dans un même état symbolique.*
   La détermination de cette nouvelle horloge se fait en fonction de la propriété que l'on veut vérifier (ici c'est la probabilité de réussite de l'envoi d'un message dans un délai de 15 *unité de temps).*

4. *Enfin il convient de réaffecter à chaque transition probabiliste sa probabilité, en fonction de sa probabilité dans l'automate temporisé probabiliste initial.*

Nous obtenons donc (à la main dans ce cas simple) où en utilisant des model checkers temporisés classiques (HyTech ou Kronos par exemple) l'automate suivant, en attendant d'obtenir un point fixe après itération des opérateurs de successeurs par transition et de successeur temporels. L'automate obtenu dans notre cas est donné en figure 4.2

Dans cet exemple simple, l'automate obtenu est une chaîne de Markov, nous pouvons donc utiliser ou PRISM (ou d'autres model-checkers probabilistes comme APMC) pour calculer les probabilités d'accessibilité (avec les méthodes données dans les chapitres précédents). ♣

4.2.3 Graphe des régions

La seconde approche pour la vérification des automates temporisés probabilistes est l'utilisation du bien connu graphe des régions. Un tel graphe est obtenu d'une partition finie de l'espace des valuations d'horloges définie dans le contexte des automates temporisés classiques [AD94]. L'extension du graphe des régions aux automates temporisés probabilistes est très détaillée dans [KNSS02]. Il suffira ensuite d'étudier l'automate des régions (un processus de décision markovien) pour vérifier les propriétés à l'aide de PRISM par exemple.
Nous donnons ici les principales étapes de constructions du graphe des régions, sans détailler. Les détails, et les résultats théoriques sont donnés dans [AD94] pour les automates temporisés classiques, et dans [KNSS02] pour les automates temporisés probabilistes.

1. On calcule la plus grande constante à laquelle sont comparées les horloges, à la fois dans l'automate et dans la propriété que l'on veut vérifier
2. On définit une notion d'équivalence entre les couples états-valuations d'horloges et on appelle les régions augmentées les classes d'équivalence pour cette relation (l'adjectif augmentée est du au fait qu'on considère aussi la propriété).
3. On construit le graphe des régions ayant pour états les régions augmentées et pour transitions les transitions probabilistes ou temporelles déterminables à partir de l'automate temporisé probabiliste d'origine. Ce graphe des régions est un processus de décision markovien.
4. On vérifie la propriété grâce aux techniques évoquées dans les chapitres précédents sur les processus de décision markoviens.

La technique d'analyse par graphe des régions est la plus générale et la plus classique, mais elle est souvent difficile et lourde à mettre en place. C'est pour cela que nous n'avons pas choisi de l'utiliser dans les exemples que nous avons étudiés.

4.2.4 Sémantique entière

La dernière technique utilisée est celle de la sémantique entière. Rappelons qu'elle nécessite que l'automate d'origine soit fermé et sans garde diagonale. La définition de cette sémantique est donnée dans la définition 4.2.4. Dans ce cas, le temps passe de manière discrète et les horloges ne sont plus que des variables comme les autres dans le processus de décision markovien. Le plus simple dans ce cas est de détailler la méthode pour notre petit exemple qui est évidemment fermé et sans garde diagonale.

Exemple 4.12. *Dans l'exemple de la figure 4.1, le graphe (cette fois-ci un processus de décision markovien) obtenu en considérant la sémantique entière est simple. Malgré tout, il faut quand même fixer une deadline (en rapport avec les propriétés que l'on souhaite vérifier), afin de rendre cet automate fini en rajoutant une nouvelle horloge (qui ne sera jamais remise à zéro) pour mesurer le temps écoulé depuis le début.*
De cette manière (et en prenant une deadline à 12 *unités de temps) on obtient le graphe de la figure 4.3 :*

Nous pouvons remarquer que cette technique est souvent la plus efficace dans le cas des automates temporisés probabilistes fermé et sans garde diagonales. En effet, il est très facile de

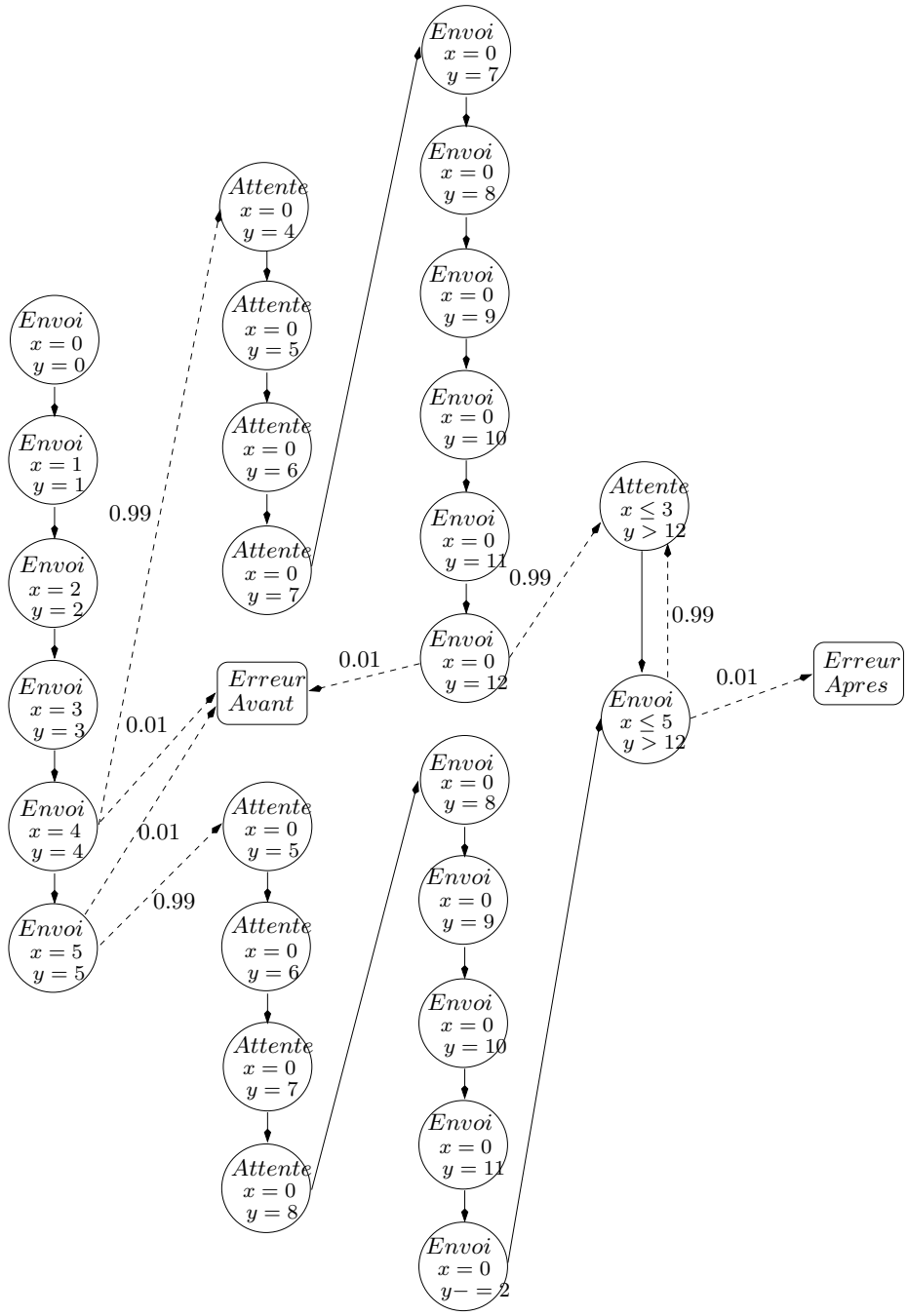


FIG. 4.3 – Graphe obtenu par sémantique entière

construire le graphe obtenu par sémantique entière et il est aussi souvent très simple. dans notre cas on peut se passer d'un grand nombre d'états (si la propriété à vérifier concerne l'échec) car les passages entre eux sont déterministes. ♣

Chapitre 5

Etude de cas : CSMA/CD

Comme je n'étudiais rien j'apprenais beaucoup.
Anatole France

Ce chapitre sera consacré à l'application des méthodes développées dans le chapitre précédent à une étude de cas menée à bien dans le cadre du projet RNTL Averroes [Ave]. Tous les résultats et les expériences que nous allons décrire ici se trouvent dans deux articles [FMP04d, DFH+04]. Nous reprendrons donc les notations utilisés dans ces deux articles.
Le protocole CSMA/CD (en anglais, Carrier Sense Multiple Access/ Collision Detection) est utilisé pour la transmission de données dans les réseaux Ethernet (Standard international IEEE 802.3 [IEE]). Ce protocole est probabiliste dans le but d'éviter au maximum les collisions entre les messages.
Ce chapitre sera séparé en deux parties, la première dans laquelle nous étudierons une version paramétrée puis probabiliste du modèle de Nicollin Sifakis et Yovine [NSY92], en utilisant les model-checkers HyTech et PRISM, et en suivant les résultats de [FMP04d] ; la seconde, sera quant à elle consacrée à une modélisation plus élaborée détaillée dans [DFH+04] et qui nous a permis d'avoir une grand nombre de résultats quantitatifs (en utilisant les deux model checkers PRISM et HyTech [PRI, HyT]).

5.1 Introduction

Dans le protocole Ethernet, plusieurs stations (NIC) peuvent se connecter sur le même canal. Comme deux stations peuvent envoyer des messages simultanément, des collisions peuvent se produire, qui détruiront les deux messages. Ainsi, les stations reçoivent en même temps un avis de collision, mais ne peuvent pas réémettre directement sous peine d'engendrer une nouvelle collision. De ce fait, le protocole CSMA/CD impose aux stations d'attendre un temps aléatoire avant de réemettre leur message.
Nous allons dans cette partie donner une description plus détaillée du protocole puis décrire en quoi notre travail est différent de ceux (nombreux) faits auparavant sur le même sujet.

5.1.1 Description du protocole

Le protocole CSMA/CD est un protocole qui gère les flux de communications entre plusieurs stations (que nous appellerons parfois émetteurs), qui dialogue par un canal unique.

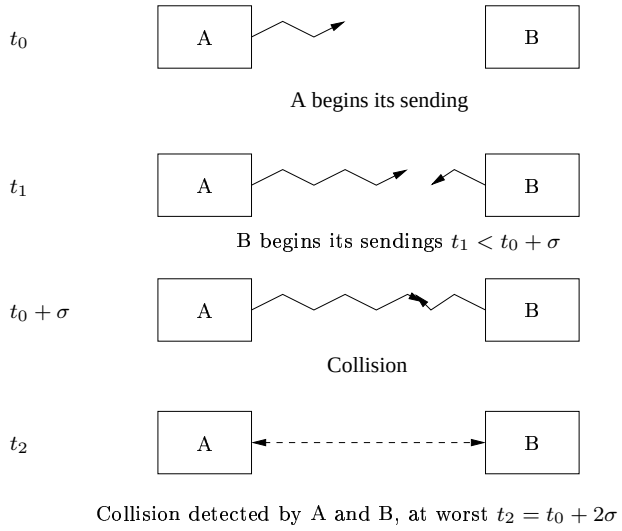


FIG. 5.1 – Une collision

La description entièrement détaillée de ce protocole se trouve dans les standards IEEE [IEE]. Nous nous sommes uniquement intéressés à la version *half duplex* de ce protocole, ce qui implique qu'un seul message peut se déplacer dans le canal à la fois (en opposition à *full duplex*). La description que nous donnons ici ne détaille pas les points dont nous n'aurons pas besoin tels que la structure des messages, ou autres.

Emission : Toutes les stations peuvent de manière identique envoyer des messages sur le réseau (Multiple Access). Chaque station doit d'abord écouter le canal pour savoir si il est libre ou occupé (Carrier Sense), et attendre que le canal apparaisse comme libre pour envoyer son message. Comme un message prend un temps borné (on le notera σ) pour traverser le canal, deux agents (ou plus) peuvent en même temps estimer que le canal est libre et envoyer leur message (pratiquement) simultanément, ce qui engendre une collision et ensuite un message de collision. Ainsi, au bout de 2σ μs au pire, les stations se rendent compte que leur message n'a pas été transmis correctement.
Après ce temps, si aucune collision n'a été détectée, la station peut, de manière certaine, terminer d'émettre. Le temps pour une émission complète est supposé constant et sera noté λ. Par exemple, pour une communication par Ethernet 10 Mbps avec $\sigma = 24\mu s$, nous avons un temps de transmission de $780\mu s$ pour un message de 1024 bits.

Conflit : En cas de collision, les messages se perdent et les différentes stations sont informés de l'événement par la réception d'un message brouillé. Elles choisissent alors, indépendamment, et de manière aléatoire un temps d'attente avant d'essayer de renvoyer le message. Pour minimiser la probabilité d'obtenir d'autres collisions, le temps d'attente est choisi uniformément dans un intervalle $[0, 2^m]$, où m représente le minimum entre α, une constante du protocole, et le nombre de collisions précédemment détectées. Ainsi, intuitivement plus le nombre de collisions est important, plus l'intervalle est grand et donc plus les chances de générer une nouvelle collision sont faibles.
Un comportement menant à une collision est décrit en figure 5.1

5.1.2 Travaux connexes

Le protocole CSMA/CD a été grandement étudié en utilisant différentes techniques. Nous nous intéresserons, dans cette partie sur les techniques basées sur le model checking, et l'étude des propriétés quantitatives.
Les études précédentes concernant CSMA/CD se sont principalement portées sur l'évaluation des performances utilisant deux approches : les modèles analytiques [GT88, TK85] ou la simulation [WK99]. Plusieurs modèles différents pour analyser à la fois la capacité du réseau et le temps d'attente, depuis les modèles traditionnels [TK85] aux plus complexes et évolués [Ber99, Kat93]. D'autres auteurs mesurent les performances en utilisant des simulations précises ce qui permet d'éviter les hypothèses simplificatrices nécessaires dans les modélisations [BMK88].

Il n'existe que très peu d'articles relatant de la vérification automatique des spécifications temporelles et probabilistes du protocole CSMA/CD. Dans [Han91], les auteurs donnent des contraintes temporelles à l'aide d'un système discret. Par contre, dans [DOTY96], le comportement du système est décrit par un produit synchronisé d'automates temporisés, puis le model checker temporisé KRONOS [KRO] est utilisé pour vérifier des propriétés telles que :

- Une collision est détectée lorsque deux stations envoient simultanément un message.
- Une collision est détectée en moins d'un certain temps.
- Quand une station commence à émettre, il existe une éxécution permettant de faire terminer l'envoi.

Mais, les model checkers temporisés classiques comme UPPAAL [UPP] et KRONOS [KRO] ne permettent pas de vérifier la satisfaction des spécifications probabilistes. Le protocole est donc modélisé en remplaçant les transitions probabilistes en des transitions non-déterministes.

Enfin, dans [KNS02], les auteurs utilisent le model checker probabiliste PRISM [PRI] pour vérifier les propriétés probabilistes du protocole CSMA/CA (IEEE 802.11 [IEE]. Par exemple, ils calculent la probabilité minimale pour les deux stations d'envoyer leur message correctement, et la probabilité minimale qu'un message soit envoyé avant une certaine deadline.

Nous nous sommes donc intéressés à deux questions peu étudiées pour le moment, la première [FMP04d] est l'étude paramétrée (en utilisant le model checker HyTech [HyT]) et probabiliste (avec PRISM) du protocole modélisé dans [DOTY96]. La seconde est l'étude probabiliste (avec PRISM) et approchée (avec APMC [APM]) de ce même protocole suivant un modèle que nous avons nous-mêmes élaboré [DFH+04].

5.2 Vérification du modèle paramétré de Nicollin-Sifakis-Yovine

5.2.1 Adaptation du modèle de Nicollin-Sifakis-Yovine

Le modèle de CSMA/CD élaboré par Nicollin, Sifakis et Yovine [NSY92] comporte deux stations émettrices reliés par un canal supposé sans faille. Seuls, les messages en cours d'envoi sont modélisés, on ne peut rien garder en mémoire.
Le protocole est la composition parallèle des deux stations et du canal. Les actions $free$ et $busy$ sont utilisés pour décrire la manière dont le canal est ressenti par les stations, elles mêmes

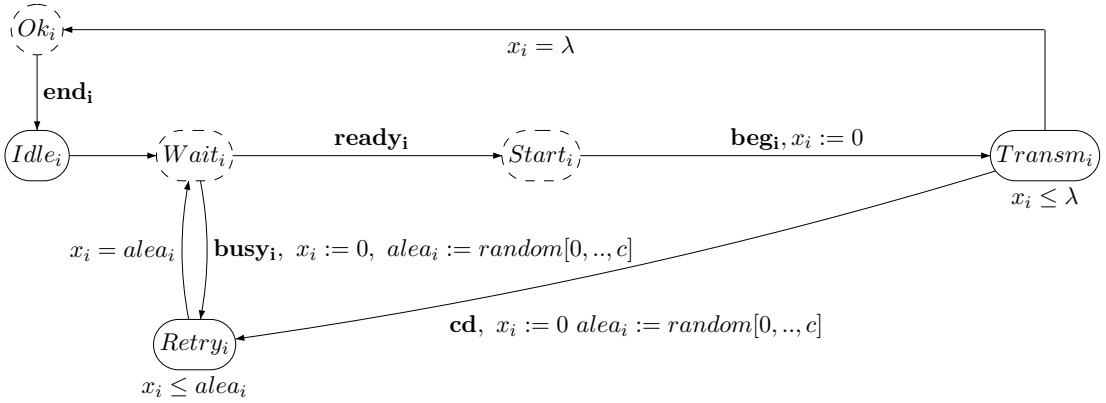


FIG. 5.2 – Automate d'une station.

idle ou *transmitting*. Le départ (resp. la fin) d'une transmission par la $i^{\text{ème}}$ station est indiqué par l'action (synchronisée avec le canal) beg_i (resp end_i). Enfin, une collision détectée est décrite par l'action cd synchronisée aussi avec le canal.

On peut donc ainsi décrire le comportement du canal. Initialement, il est vide et peut accepter un message de chacune des stations. Supposons qu'une des stations commence à envoyer, il existe un intervalle de temps de longueur σ pendant lequel le canal acceptera l'envoi d'un message par l'autre station, ce qui causera une collision. Si aucune collision n'apparaît, le canal attendra la fin du message. Il attend ensuite σ (temps pour que le canal se vide) et retourne dans l'état initial. On peut remarquer qu'il y a donc un délai minimum entre deux messages consécutifs envoyés (de σ) dû au délai de propagation du canal.

En suivant la méthode décrite dans [KNS02], nous avons adapté le modèle pour le décrire sous la forme d'un automate temporisé probabiliste. L'aspect probabiliste apparaît dans le choix du délai d'attente après une collision (dans un intervalle $[0, c]$ avec $c = 2^k\sigma$, mais que nous considérerons ici comme un paramètre). Le modèle de CSMA/CD est donc composée de trois parties (chacune étant représenté par des automates temporisés probabilistes) : $Station_1$, $Station_2$, $Canal$. Dans le produit synchronisé $Station_1||Station_2||Canal$, l'ensemble $\mathcal{X}$ des horloges du système est $\{x_1, x_2, y, t\}$ où x_i représente l'horloge de la $Station_i$, y celle du $Canal$ et t une horloge globale permettant de calculer le temps écoulé depuis le début de la session. Une configuration de l'automate produit est de la forme $((l_1, l_2, l_3), \overline{v})$ dans laquelle l_i $(i = 1, 2)$ est l'état de l'automate $Station_i$, l_3 est l'état du $Canal$ et $\overline{v}$ est une valuation des différentes horloges de $calX$. Nous pouvons à présent décrire les deux automates $Station_i$ et $Canal$.

Automate $Station_i$ **:** Cet automate décrit le comportement de chacune des deux stations. Il contient une horloge x_i et une variable contenant le temps d'attente aléatoire $alea_i$. L'automate est décrit dans la figure 5.2.

Il y a six états, parmi lesquelles ok_i, $wait_i$ et $Start_i$ (représentés par des cercles pointillés) sont urgentes, dans lesquels le temps ne passe pas. L'état initial est $Idle_i$. Après être resté un certain temps dans cet état (cette durée étant non-déterministe et représentant l'envie ou non d'envoyer un message), la station veut émettre un message et se retrouve dans l'état transitoire $Wait_i$, dans lequel le canal est écouté. Si le canal semble libre (action $ready_i$),

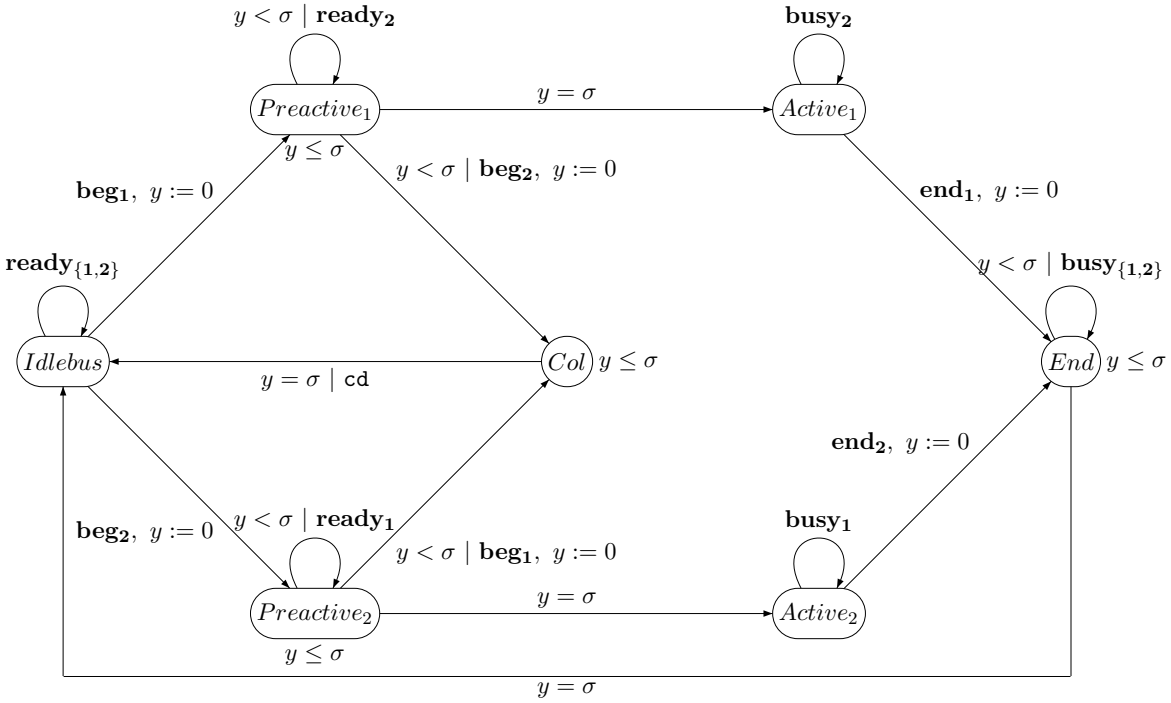


FIG. 5.3 – Automate du canal.

la station commence son envoi, en se rendant dans l'état $Transm_i$ (action beg_i). Dans le cas contraire, (action $busy_i$), il se retrouve dans l'état $Retry_i$, dans lequel il reste pendant le temps d'attente aléatoire $alea_i$ avant de retourner dans l'état $Wait_i$. Dans l'état $Transm_i$, la station doit attendre λ, après quoi elle émet un signal annonçant la fin du message (action end_i) et retourne dans l'état $Idle_i$ après un passage par l'état transitoire Ok_i. De plus, une collision peut se produire (action cd), forçant la station à arrêter son émission, quittant son état $Transm_i$ pour l'état $Retry_i$ dans lequel elle patiente pendant une période aléatoire $alea_i$. La valeur maximale d'$alea_i$ est le paramètre c.

Automate *canal* **:** L'automate temporisé probabiliste $Canal$, qui décrit le comportement du canal, est représenté dans la figure 5.3.

Il y a sept états. Dans l'état initial $Idlebus$, le canal est prêt pour la transmission de messages. Si une des stations, disons la $Station_1$, commence à envoyer un message (beg_1), le canal remet son horloge y à zéro, et se rend à l'état $Preactive_1$. Tant que y est plus petit que σ le canal est toujours senti comme libre, et la $Station_2$ peut commencer à émettre un message ($beg2$). Cela entraîne alors une collision (cd). Lorsque l'horloge y atteint la valeur σ, le canal se rend dans l'état $Active_1$, et attend le signal de terminaison du message (end_1) de la $Station_1$, et est senti comme occupé *busy* pour la $Station_2$. Enfin, quand le message se termine, le canal se rend dans l'état End pendant un temps σ avant de se retrouver dans l'état initial et être en état pour transmettre à nouveau.

L'originalité de l'étude que nous avons menée dans [FMP04d] est que nous avons pu (en utilisant HyTech) étudier le protocole de manière paramétrée. En effet plusieurs constantes seront considérés dans notre cas comme des paramètres du problème :

- La durée de propagation d'un message : λ
- La durée de propagation à l'intérieur du canal : σ
- La borne supérieure pour le tirage d'*alea* : c

De plus, pour éviter des comportement limites et non intéressants, nous avons considéré $\sigma > 0$ et $\lambda \geq 2\sigma$.

5.2.2 Vérification non probabiliste utilisant HyTech

Le model-checker HyTech [HyT] permet d'étudier les propriétés des automates temporisés, avec des paramètres. Afin de l'utiliser, il faut, dans les automates $Station_i$ que nous avons présentés, changer l'affectation aléatoire $alea_i = random[o..c]$ par une affectation non déterministe d'$alea_i$ à un nombre compris entre 0 et c.

Propriété : Nous nous sommes intéressés à la propriété de vivacité suivante : lorsque une station, disons $Station_1$, a un message à envoyer, alors il existe une transmission réussie. Nous pouvons donc exprimer cette propriété dans la logique temporelle [HNSY94] TCTL par la formule suivante :

$$Init \Rightarrow \forall\Box(l_1 = Wait_1) \Rightarrow \exists \diamond\, l_1 = Ok_1$$

sans laquelle $Init$ correspond à l'état initial global du système. Dans la preuve les variables σ, λ et c seront considérés comme des paramètres vérifiant $o \leq 2\sigma \leq \lambda$ et $\sigma \leq c$.

Vérification par HyTech : Les automates du canal et des stations peuvent être directement implémentés dans *HyTech*. Les affectations non déterministes de la variable *alea* en HyTech par l'utilisation de l'instruction $alea_i' \geq 0$, et en ajoutant l'invariant $alea_i \leq c$ à l'état $Retry_i$. La borne supérieure de la variable *alea* est un paramètre fixé par l'instruction $c \geq \sigma$. De la même manière, le paramètre λ est fixé par l'instruction $2\sigma \leq \lambda$ et enfin le paramètre σ par $\sigma > 0$.
Les états transitoires $(Wait, Start, Ok)$ sont implémentés en utilisant une horloge annexe z remise à zéro dès qu'on entre dans ces états et un invariant $z = 0$. Le code complet en HyTech est donné en section 5.2.4.
L'exploration en avant (consistant à calculer tous les états accessibles à partir d'un état) par HyTech $Post*(Init)$ requiert 21 itérations, où l'état $Init$ correspond à
ℓ_1=`Idle1` & ℓ_2=`Idle2` & ℓ_3=`Idlebus` & `x1=0` & `x2=0` & `y=0`
& `lambda>=2sigma` & `c>=sigma` & `sigma>0 ;`
L'exploration en arrière (consistant à calculer tous les états menant à un état) $Pre*(final)$ requiert, quant à elle, 80 itérations. Notons que dans ce cas, il est nécessaire de fixer une borne supérieure à λ du type $k\sigma$. (80 correspond à $k = 900$).Ici $final$ est l'état défini par :
ℓ_1=`Ok1` & `lambda>=2sigma` & `c>=sigma` & `sigma>0` & `lambda<=k*sigma ;`

De cette manière on peut récrire la propriété que nous voulons vérifier sous la forme équivalente suivante :
$(Post^*(Init) \cap \{\ell_1 = Wait_1\}) \subseteq Pred^*(\ell_1 = Ok_1)$,
Et nous avons prouvé en HyTech :

$$(Post^*(Init) \cap \{\ell_1 = Wait_1\} \cap \{\lambda \leq k\sigma\}) \cap \neg(Pred^*(Final) \cap \{\ell_1 = Wait_1\}) = \emptyset.$$

Pour $k = 30$ le calcul complet a pris 1780 secondes en utilisant un PENTIUM IV de 2,80 GHz

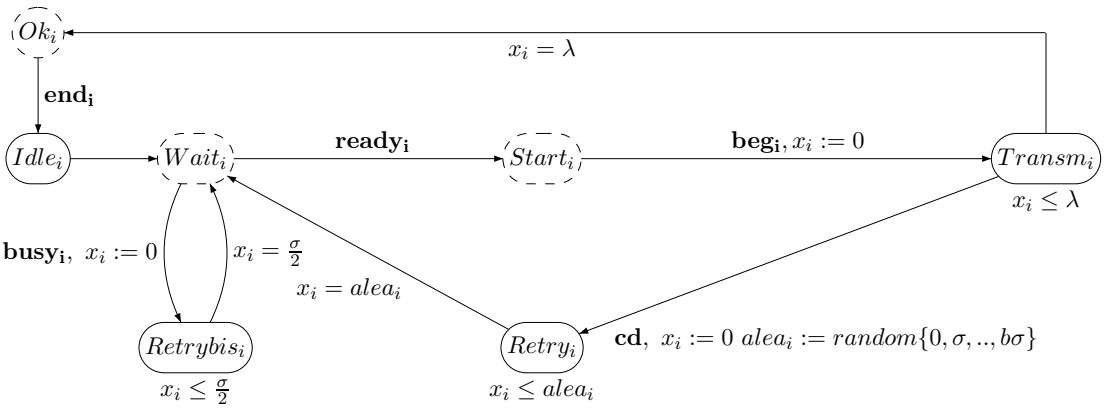


FIG. 5.4 – Nouvelle version de l'automate de la station i.

avec 1Gb de RAM.

On peut remarquer aussi que la propriété est fausse dans le cas où $\lambda < 2\sigma$. HyTech nous permet, en plus, d'exhiber une trace d'exécution correspondant au comportement anormal : Tout d'abord la $Station_2$ émet à $t = 0$, puis la $Station_1$ se rend dans l'état $Wait_1$ et commence à émettre à l'instant t', avec $\sigma > t' > \lambda - \sigma$. Il y a donc une collision à l'instant t', telle que le $Canal$ se rend à l'état Col et y et remis à zéro. Le premier message s'achève à l'instant $t'' = \sigma$, avec $t'' < t' + \sigma$, ce qui envoie le système entier à l'état $(Transm_1, Ok_2, Col)$. Maintenant, le temps ne peut plus passer, car l'action end_2 ne peut pas avoir lieu, le $canal$ n'étant pas dans l'état $Active_2$. Ce comportement correspond à un comportement Zeno, il est aisé de voir que dans le cas $\lambda \geq 2\sigma$ on évite tous les comportements Zeno.

5.2.3 Vérification probabiliste en utilisant alors HyTech et PRISM

Le model-checker HyTech nous a aussi permis de faire l'analyse en avant 4.2.2 de cet automate temporisé probabiliste afin d'obtenir un processus de décision markovien sur le graphe des zones. Pour vérifier les propriétés probabilistes, nous avons repris un modèle plus proche de celui du standard IEEE [IEE]. En effet le temps d'attente probabiliste n'est tiré qu'en cas de collision, par contre si le canal est senti occupé par l'une des stations elle attend un temps fixe (ici $\sigma/2$), avant de réessayer.

De plus par la suite nous supposerons qu'$alea$ est un multiple de σ nous le tirerons donc dans l'intervalle $[0, \sigma, ..., b\sigma]$ plutôt que dans l'intervalle complet $[0, 1, 2, .., b\sigma]$.
Le nouvelle automate représentant le comportements des stations est représentée dans la figure 5.4

Propriété :
La propriété qui nous intéresse ici, est une propriété d'accessibilité minimale :
La probabilité minimale qu'après une collision les deux stations envoient correctement leur message avant une deadline d. On peut l'écrire dans la logique LTL par la formule suivante, partant de $Init' = Retry_1.Retry_2.Idlebus$ (une collision vient juste d'apparaître) :

$$P_{min}[trueU\{F\ Ok_1 \wedge F\ Ok_2 \wedge t \leq d\}]$$

Afin de pouvoir vérifier cette propriété en utilisant le model-checker PRISM [PRI], nous avons introduit deux variables booléennes f_i qui sont incrémentés lorsque la station concernée envoie

correctement leur message. Ainsi la formule s'écrit (dans la logique PTCTL [KNSS02]) :

$$P_{min}[trueU\{f_1 = 1 \wedge f_2 = 1 \wedge t \leq d\}].$$

Dans ce cas, il est évident que le comportement le plus défavorable est celui ou chaque station a toujours envie d'envoyer un message. Plus précisément, on peut considérer que la source de non déterminisme du système peut être retirée en considérant que l'état *Idle* est transitoires (dans le sens où le temps ne passe pas dans cet état). Nous aurons (une fois le graphe des zones obtenus) une chaîne de Markov à étudier.

Analyse en avant par HyTech :

Suivant les méthodes décrites en [KNS02, KNS, IEE] et dans le paragraphe 4.2.2, nous allons vérifier la propriété par analyse en avant de l'automate temporisé probabiliste. Ainsi nous avons construit l'automate de la figure 5.5 en utilisant les zones obtenues par HyTech et la méthode précise de [KNS02, KNS]. L'état *GOOD* a été rajouté et correspond aux états dans lesquels $f_1 = f_2 = 1$.
Un résultat original de notre étude est que σ et λ sont toujours des paramètres, de plus le graphe nous fait remarquer que le système ne dépend que du ratio λ/σ.

Résultats avec PRISM :
Nous pouvons donc maintenant utiliser les algorithmes de model-checking sur le graphe des états symboliques obtenus, et nous vérifions la formule suivante (dans la logique PCTL) :

$$P_{=?}[true\ U\ GOOD].$$

Les graphes suivants 5.6, 5.7 représentent l'évolution de cette probabilité en fonction dans la deadline pour différents rapports λ/σ.

Ces figures nous permettent de voir un premier seuil à partir duquel la probabilité minimale d'envoyer les messages correctement est non nul. Après ce seuil, la probabilité augmente très rapidement et asymptotiquement vers 1. Ce qui correspond bien au but du protocole puisque la probabilité d'envoyer correctement les messages après une collision est forte.

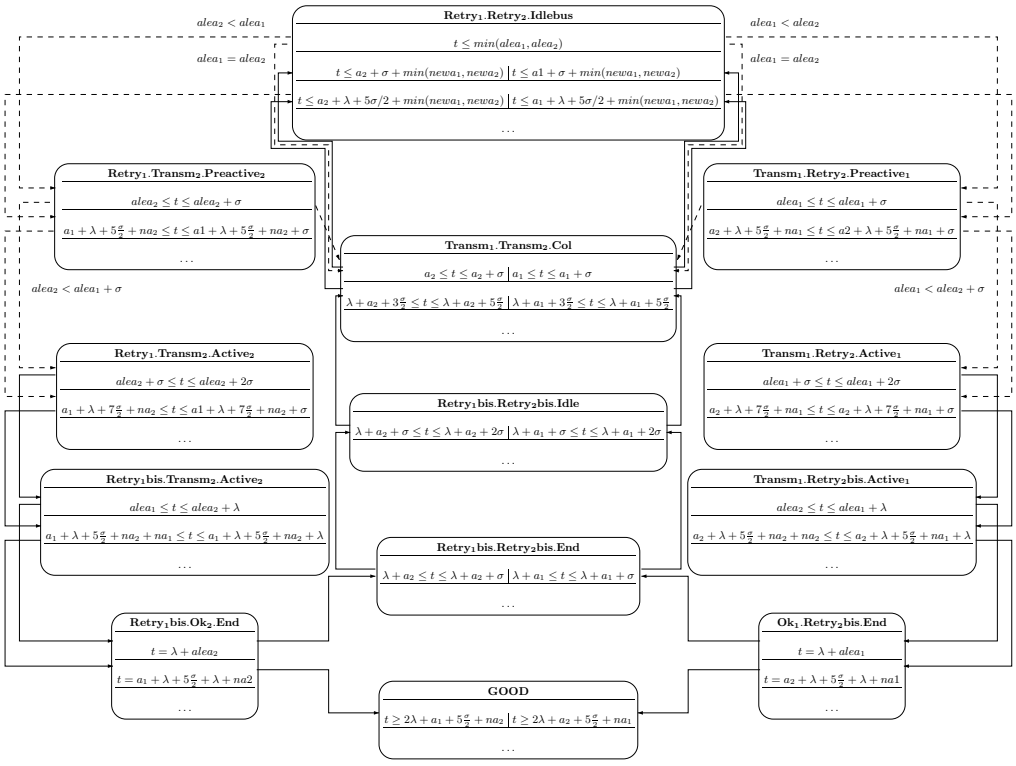
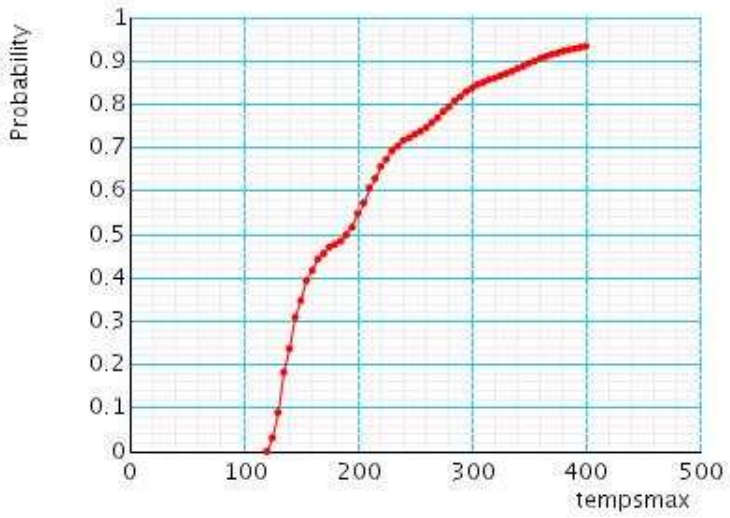
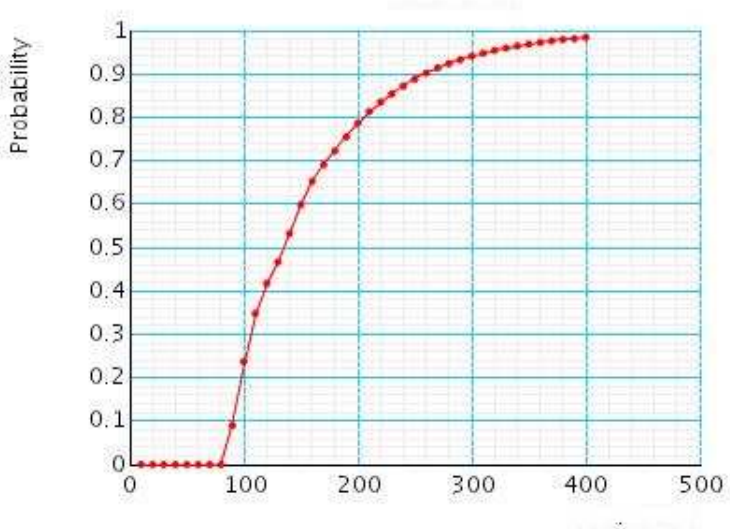


FIG. 5.5 – graphe d'accessibilité



($\lambda/\sigma = 30$, $alea_{max} = b = 3$)
FIG. 5.6 – Probabilité de l'émission en fonction du délai.



($\lambda/\sigma = 20$, $alea_{max} = b = 3$)
FIG. 5.7 – Probabilité de l'émission en fonction du délai.

5.2.4 Code HyTech

```
   var
x1,x2,y,z,t : clock;
alea1,alea2 : discrete;
lambda, sigma, c : parameter;

   - ---------------------------------
automaton Sender1
synclabs : beg1,cd,end1,ready1,busy1;

   initially Idle1;

   loc Idle1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 wait {}
when True do {z'=0} goto Wait1;

   loc Wait1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & z=0 wait {}
when True sync ready1 do {z'=0} goto Start1;
when True sync busy1 do {x1'=0, alea1'>=0} goto Retry1;

   loc Start1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & z=0 wait
when True sync beg1 do {x1'=0} goto Transm1;

   loc Retry1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & x1 <= alea1 & alea1 <= c wait {}
when x1=alea1 do {z'=0} goto Wait1;

   loc Transm1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & x1 <= lambda wait {}
when True sync cd do {x1'=0, alea1'>=0} goto Retry1;
when x1=lambda do {z'=0} goto Ok1;

   loc Ok1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & z=0 wait {}
when True sync end1 goto Idle1;

   end - Sender1

   - ---------------------------------
automaton Sender2
synclabs : beg2,cd,end2,ready2,busy2;

   initially Idle2;

   loc Idle2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 wait {}
when True do {z'=0} goto Wait2;

   loc Wait2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & z=0 wait {}
```

```
when True sync ready2 do {z'=0} goto Start2;
when True sync busy2 do {x2'=0, alea2'>=0} goto Retry2;

   loc Start2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & z=0 wait
when True sync beg2 do {x2'=0} goto Transm2;

   loc Retry2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & x2 <= alea2 & alea2 <= c wait {}
when x2=alea2 do {z'=0} goto Wait2;

   loc Transm2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & x2 <= lambda wait {}
when True sync cd do {x2'=0, alea2'>=0} goto Retry2;
when x2=lambda do {z'=0} goto Ok2;

   loc Ok2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & z=0 wait {}
when True sync end2 goto Idle2;

   end - Sender2

   ------------------------------------
automaton Chan
synclabs : beg1,beg2,cd,end1,end2,ready1,ready2,busy1,busy2;

   initially Idlebus;

   loc Idlebus :
while x1>=0 & x2>=0 & y>=0 & t>=0 wait {}
when True sync ready1 goto Idlebus;
when True sync ready2 goto Idlebus;
when True sync beg1 do {y'=0} goto Preactive1;
when True sync beg2 do {y'=0} goto Preactive2;

   loc Preactive1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & y<= sigma wait {}
when y<sigma sync ready2 goto Preactive1;
when y=sigma goto Active1;
when y<sigma sync beg2 do {y'=0} goto Col;

   loc Preactive2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 & y<= sigma wait {}
when y<sigma sync ready1 goto Preactive2;
when y=sigma goto Active2;
when y<sigma sync beg1 do {y'=0} goto Col;

   loc Col :
while x1>=0 & x2>=0 & y>=0 & t>=0 & y<= sigma wait {}
when y = sigma sync cd goto Idlebus;

   loc Active1 :
while x1>=0 & x2>=0 & y>=0 & t>=0 wait {}
```

```
when True sync busy2 goto Active1;
when True sync end1 do {y'=0} goto End;

   loc Active2 :
while x1>=0 & x2>=0 & y>=0 & t>=0 wait {}
when y<sigma sync busy1 goto End;
when y<sigma sync busy2 goto End;
when True sync busy1 goto Active2;
when True sync end2 do {y'=0} goto End;

   loc End :
while x1>=0 & x2>=0 & y>=0 & t>=0 & y<= sigma wait {}
when y = sigma goto Idlebus;

   end - Chan

   - ----------------------------------
- analysis commands

   var init, predit, postit, final : region;

   init :=
loc[Sender1]=Idle1 & loc[Sender2]=Idle2 & loc[Chan]=Idlebus
& x1=0 & x2=0 & y=0 & t=0
& lambda >= 2*sigma & c>=sigma & sigma > 0;

   final :=
loc[Sender1]=Ok1& lambda>= 2*sigma & c>=sigma & sigma>0
& lambda <= 30*sigma;

   prints "iterated succ";
postit := (reach forward from init endreach) & loc[Sender1]=Wait1;
print hide t,z in postit endhide;

   prints "iterated pred";
predit := (reach backward from final endreach) & loc[Sender1]=Wait1;
print hide t,z in predit endhide;

   prints "INTERSECTION EMPTY?";
if empty(postit & lambda<= 30*sigma & ¬(predit))
then prints "OK";
else
print hide t,z in (postit & ¬(predit)) endhide;
endif;
```

5.3 Vérification d'un modèle élaboré

La deuxième partie de ce chapitre est consacrée à une étude de CSMA/CD en le modélisant de manière discrète [DFH+04] afin d'utiliser la sémantique entière définie en section 4.2.4. Le but est d'étudier un grand nombre de propriétés quantitatives dans un modèle qui ressemble le plus possible à celui des normes IEEE [IEE] et qui soit à la fois clos et sans garde diagonale

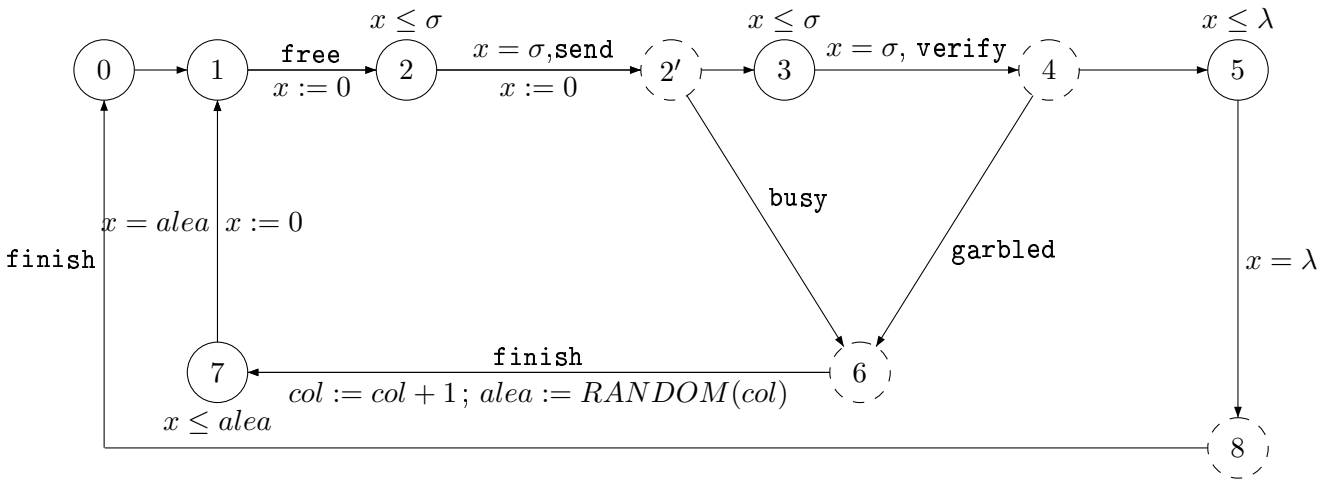


FIG. 5.8 – Automate d'une station.

(condition de validité de la sémantique entière du théorème 4.9.
De plus, pour réduire le temps de calcul des model-checkers nous avons choisi de rentrer un peu plus dans le détail (afin de réduire les synchronisations, et les horloges).
Enfin, nous avons utilisé deux model-checkers (PRISM [PRI] et APMC [APM]) qui nous ont permis d'avoir des résultats complémentaires.

5.3.1 Modélisation

De la même manière que dans les sections précédentes, nous avons modélisé ce protocole par la composition parallèle de trois automates temporisés probabilistes ($Station_1$, $Station_2$, $Canal$). Cette fois-ci nous avons 3 horloges x_1, x_2 et y. Nous noterons de la même manière que dans les sections précédentes les états et les configurations du système.
Enfin, à la différence du modèle précédent nous avons (afin de calculer les probabilités exactes) une variable supplémentaire col_i stockant le nombre de collisions des messages de $Station_i$ depuis le dernier envoi réussi. Un état de $Station_i$ est donc un couple $l_i = (s_i, col_i)$ dans lequel s_i est l'état de l'automate temporisé probabiliste.

Automate de la station : Cet automate (voir la figure 5.8) décrit le comportement de chaque station. Il contient une horloge x et une variable stockant le temps d'attente aléatoire $alea$.

Il y a dix états dans cet automate parmi lesquels, les états 2′,4,6 et 8 (représenté avec des cercles pointillés) sont des états urgents intermédiaires, dans lesquels le temps ne passe pas. De plus les événements $busy$, $free$ et $garbled$ sont des événements urgents pour la $Station$. L'état initial est 0. Quand une station désire envoyer un message elle passe dans l'état 1. Cette transition est précédée d'un certain temps passé dans l'état 0 et correspond à une source de non déterminisme. Si le canal semble libre ($free$), alors la station se rend dans l'état 2 dans lequel elle attend σ avant de retester le canal (dans l'état urgent 2′). S'il est occupé (l'autre station avait émis avant) alors la procédure de backoff (affectation d'un temps d'attente aléa-

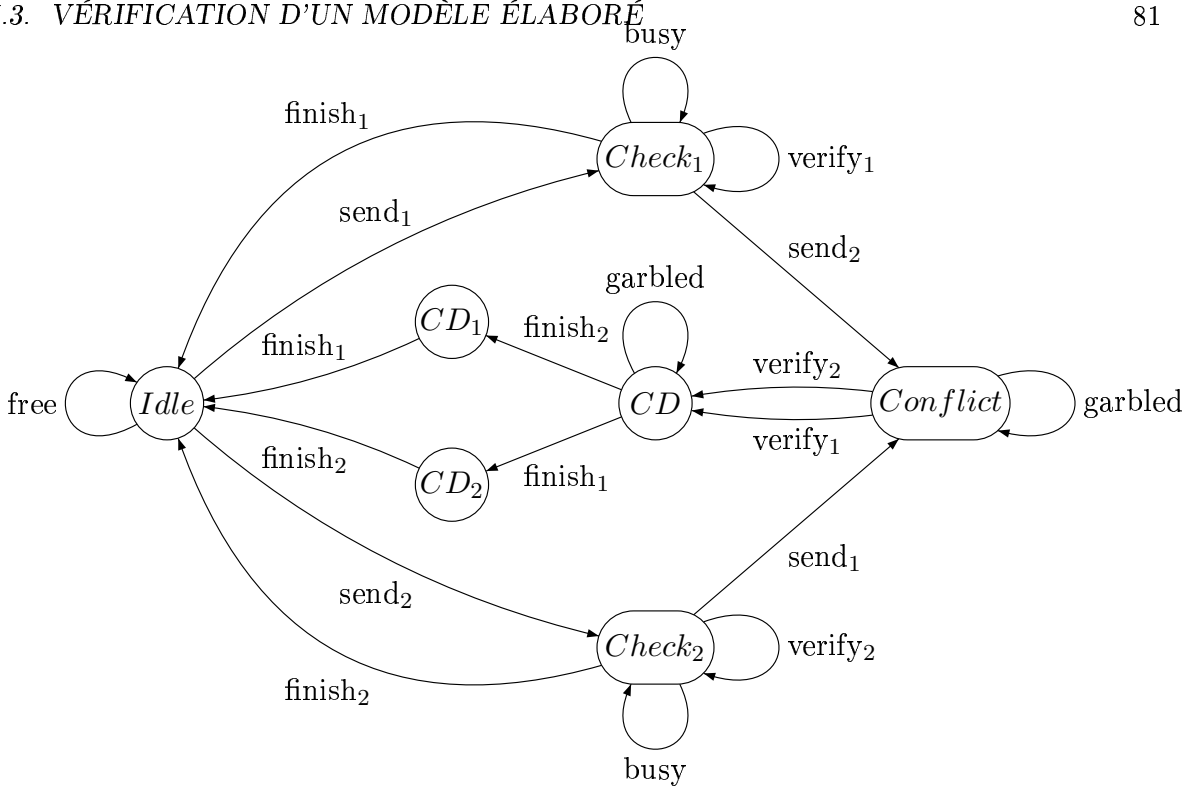


FIG. 5.9 – Automate du canal.

toire) se met en place et la station repasse par les états 6, 7 avant de retourner en 1. Si le canal n'était pas occupé, la station se rend dans l'état 3 et attend encore σ (au cas où l'autre station ait émis moins de σ après lui) et elle reteste le canal. S'il est brouillé (collision), elle réentame la procédure de backoff (6 puis 7 puis 1) sinon elle termine correctement l'envoi de son message (5 puis 8).
Rappelons que le temps d'attente aléatoire est tiré dans un intervalle $[0, 2^m - 1]$ où $m = \min(col, \alpha)$, α étant un paramètre du système.
Pour récapituler nous pouvons donner un rapide résumé des états de l'automate :

0 Inactif
1 Attend que le canal soit libre
2 Attend σ avant de tester le canal
2' Test du canal (*busy* ou non)
3 Attend σ
4 Test du canal (*garbled* ou non)
5 Termine l'envoi
6 Collision détectée
7 Attend un temps aléatoire avant la réémission
8 Message correctement envoyé

Automate du canal : L'automate du canal est un automate probabiliste (sans horloges) et est représenté en figure 5.9.

L'état initial *Idle* indique que le canal est libre et vide. Depuis cet état, la réception d'un message (événement $send_1$ par la $Station_1$) active la transition vers $Check_1$, puis se message

peut s'envoyer correctement (événement $verify_1$ puis $finish_1$) ou entrer en collision avec un message de la $Station_2$ (événement $send_2$) activant la transition vers l'état $Conflict$. Une fois la collision détectée , le canal se rend dans l'état CD et un message brouillé est envoyé (événement $garbled$). Il retourne dans l'état $Idle$ une fois reçu le message de fin de la part des stations (événement $finish$).

Propriétés : Contrairement à la partie précédente, nous nous sommes intéressés ici, à un grand nombre de propriétés quantitatives :

- Propriétés d'accessibilité minimale et maximale sur des automates temporisés probabilistes :
  - La probabilité minimale pour qu'une station envoie correctement un message avec une deadline d :
    $P_{min}[\overline{s} \rightarrow^* \{s \mid (x_1 = 8 \vee\ x_2 = 8)\ \wedge\ y \leq d\}]$
  - Les probabilités maximales et minimales pour qu'il y ait au moins N collisions avant d :
    $P_{min}[\overline{s} \rightarrow^* \{s \mid col_1 \geq N\ \wedge\ y \leq d\}]$
    $P_{max}[\overline{s} \rightarrow^* \{s \mid col_1 \geq N\ \wedge\ y \leq d\}]$
- Propriétés d'accessibilités sur des systèmes complétement probabilistes (chaîne de Markov).
  - La probabilité qu'une station envoie correctement son message avant une deadline d.
    $Prob[\overline{s} \rightarrow^* \{s \mid (x_1 = 8 \vee\ x_2 = 8)\ \wedge\ y \leq d\}]$
  - La probabilité qu'il y ait au moins N collisions avant d :
    $Prob[\overline{s} \rightarrow^* \{s \mid col_1 \geq N\ \wedge\ y \leq d\}]$

5.3.2 Vérification par PRISM

Rappelons que nous avons choisi d'utiliser la sémantique entière pour des raisons de simplicité (pas de graphes des régions à construire) et d'efficacité. Les horloges sont donc considérées comme des compteurs et représentés par des variables entières. Ainsi l'automate temporisé probabiliste peut-être étudié comme un processus de décision markovien (voir chapitre 4).
Rappelons aussi, que nous avons, en plus des propriétés citées dans la section précédente calculer les espérances (minimale et maximale) du temps pour qu'un envoi soit réussi.

Résultats

Toutes les expériences ont été faites sur un PENTIUM IV 2.8 GHz avec 1 Gb de RAM. La taille des modèles engendrés (jusque douze millions d'états) nous a fait choisir la construction par MTBDD disponible dans PRISM, qui est beaucoup plus efficace pour les gros systèmes.
La différence majeure avec APMC (voir la section 5.3.3) est que PRISM peut manier le non déterminisme inhérent au modèle. Il construit effectivement le modèle et trouve le pire et le meilleur des cas pour chacune des propriétés. On peut ainsi laisser le choix d'envoyer ou non un message comme un choix non déterministe.
La taille du modèle nous a rendu impossible (avec PRISM) la vérification avec les constantes réelles λ et σ. Par contre nous avons conservé le rapport entre ces deux constantes. Nous avons aussi réduit un peu la constante α (utilisé pour calculer l'intervalle de tirage de la variable aléatoire $alea$) de 10 à 6.

Les trois principales propriétés que nous avons vérifiées sont les suivantes :

- La première propriété représente la probabilité maximale pour atteindre un état dans lequel au moins une des deux stations a envoyé correctement avant la deadline d. On peut l'écrire dans la syntaxe de PCTL par la formule suivante :

$$P_{min}[true\ U((s_1 = 8|s_2 = 8)\ \&\ y \leq d)]$$

- Les deux autres propriétés représentent les probabilités minimales et maximales pour atteindre un état dans lequel N collisions se sont produites pour une station fixée depuis son dernier envoi réussi et avant une deadline d. Elles s'expriment par les formules suivantes :

$$P_{min}[true\ U(col_1 = N\ \&\ y \leq d)]$$
$$P_{max}[true\ U(col_1 = N\ \&\ y \leq d)]$$

La figure 5.10 donne la probabilité de satisfaire la première propriété pour différentes valeurs de d. Il y a trois courbes correspondant au degré de non déterminisme laissé dans le système. En effet il y a plusieurs sources de non déterminisme, la première vient du fait que la station peut attendre un temps quelconque dans l'état 0 avant de vouloir émettre un message (bien sûr, nous supposons qu'une des deux stations veut émettre sinon la probabilité minimale serait toujours nulle). L'autre source de non déterminisme vient de l'ordre des actions entre les deux stations (comme $send_1$ et $send_2$). La première courbe correspond au modèle originel, la seconde est obtenu en remplaçant la transition non déterministe de 0 à 1 par une transition probabiliste, et la dernière en compilant le système comme un système totalement probabiliste. Notons que cette dernière courbe est exactement la même que celle obtenu par APMC 5.15. Les courbes étant similaires, l'analyse sera faite dans la section suivante.

La figure 5.11 nous donne les résultats pour les autres propriétés (nombre de collisions). Nous avons choisi une deadline de 210 car avec cette deadline la probabilité qu'un message soit délivré est très grande (plus de 0,98). De la même manière nous avons construits les courbes pour les trois types de modèles différents. Remarquons que la probabilité d'avoir 3 collisions est très importante, mais décroˆtrès rapidement pour devenir faible pour la constante $\alpha = 6$ que nous avons choisi.

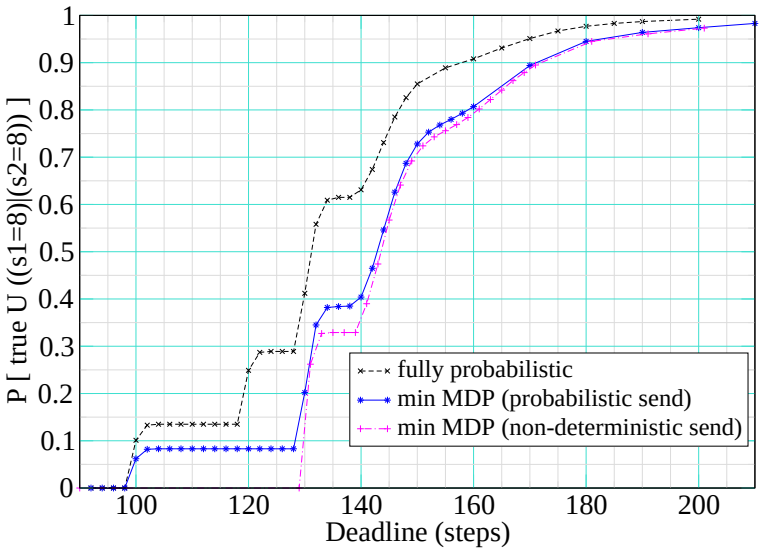
De plus nous avons calculer l'espérance de temps minimale et maximale pour envoyer un message, déterminé par la formule suivante :

$R_{max}[true\ U((s_1 = 8)|(s_2 = 8))]$

Les résultats sont de 147 pour le modèle avec l'envoi non déterministe, et de $144,4$ pour celui avec l'envoi probabiliste, il n'y a pas beaucoup de différences entre ces deux résultats.

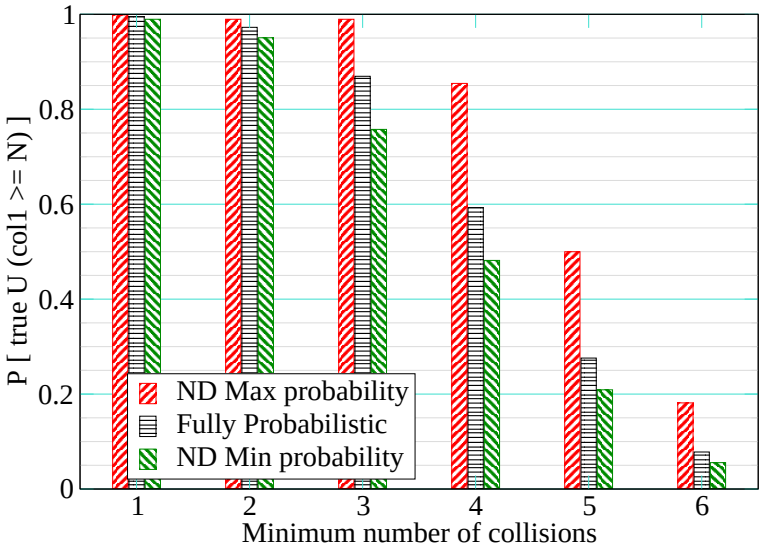
Nous avons aussi calculé le temps moyen pour qu'une station donnée envoie correctement son message :

$R_{max}[true\ U(s_1 = 8)]$
Par contre dans ce cas, la présence de non déterminisme ou non est cruciale. Les résultats



($\lambda = 96$, $\sigma = 3$, $\alpha = 6$)

FIG. 5.10 – Probabilité d'émission en fonction du délai



($\lambda = 96$, $\sigma = 3$, $\alpha = 6$)

FIG. 5.11 – Probabilité maximale et minimale d'avoir au moins N collisions

sont de 707 pour le modèle avec l'envoi probabiliste et de 5750 pour le non déterministe. Cela montre qu'une station peut retarder l'envoi d'un message par l'autre station en envoyant continûment des messages et en chargeant le canal. De plus, ce résultat vient du fait que si une station réussit à envoyer un message son compteur de collision est remis à zéro mais pas celui de l'autre station, elle attend donc des long temps avant de pouvoir réenvoyer.
Cette dernière propriété montre l'importance d'avoir étudier un système probabiliste et non déterministe (grâce à PRISM).

5.3.3 Vérification par APMC

APMC est un model-checker probabiliste approché dédié uniquement à la vérification des propriétés quantitatives sur les chaînes de Markov à temps discret. Le langage d'entrée est le même que celui de PRISM ce qui nous a permis d'exporter notre modèle aisément. A partir de maintenant, nous nous plaçons dans le cas où toutes les sources de non déterminisme ont été retirées et la similarité de plusieurs résultats comme ceux des figures 5.10 et 5.11 nous autorise à étudier ce modèle avec légitimité. De plus les méthodes utilisés pour le model-checking dans APMC vont nous permettre de manier des modèles beaucoup plus gros et donc d'instancier les automates avec les constantes réelles pour σ, λ et α.

Fondements théoriques de APMC

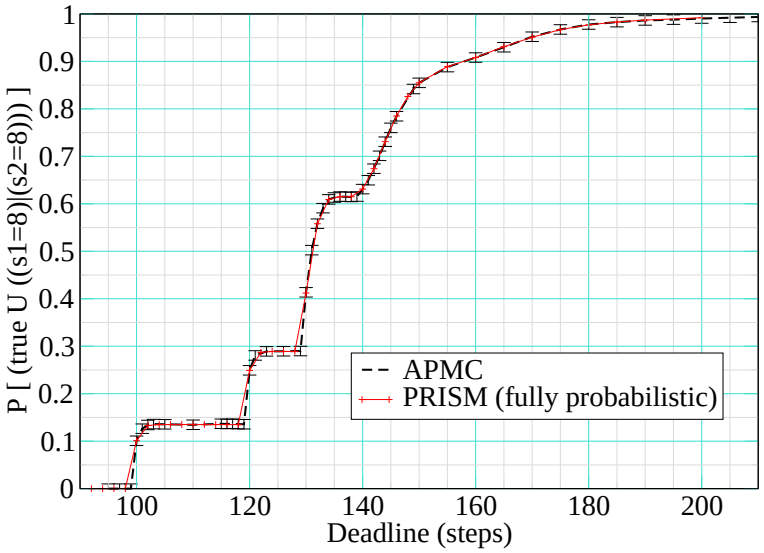
La technique utilisée dans APMC [HLMP04] utilise une méthode de Monte-Carlo efficace pour approcher les probabilités de satisfaction de propriétés monotones sur des chaînes de Markov. Les propriétés vérifiées sont exprimées dans la logique temporelle : LTL. De plus, nous utiliserons les notations du chapitre 2 pour décrire les chaînes de Markov. Enfin nous noterons $Path(s)$ tous les chemins commençant en s et $Path_k(s)$ ceux de longueur k, ainsi $\mathbb{P}_k(\phi)$ représentera la probabilité d'une formule ϕ sur les chemins de longueurs k.

Définition 5.1. *Une formule LTL ϕ est dite monotone si et seulement si pour tout $k > 0$, pour tout chemin π de longueur k, $\mathcal{M}, \pi \models \phi \Rightarrow \mathcal{M}, \pi^+ \models \phi$, où π^+ représente un chemin quelconque ayant pour préfixe π.*

Une propriété élémentaire des formules monotones est que : Si ϕ est une formule monotone, et $0 < b \leq 1$ et si il existe $k \in \mathbb{N}$ tel que $\mathbb{P}_k[\phi] \geq b$ alors $\mathbb{P}[\phi] \geq b$.
De ce fait, afin de vérifier une spécification probabiliste de la forme $\mathbb{P}[\phi] \geq b$, nous choisissons d'abord une valeur de $k = O(log|S|)$, et nous calculons de manière approchée $\mathbb{P}_k[\phi]$ pour tester si sa valeur est plus grande que b. Si tel est le cas, la monotonie de la formule nous donne le résultat. Sinon, nous incrémentons la valeur de cas et calculons à nouveau $\mathbb{P}_k[\phi]$. Nous itérons cette procédure en bornant la valeur maximale de k qui, dans de nombreux cas, est logarithmique en le nombre d'états du système. Si les résultats de tous les tests sont négatifs nous concluons que $\mathbb{P}[\phi] < b$.La manière dont nous approchons la valeur de $\mathbb{P}_k[\phi]$ utilise les FPRAS (fully polynomial randomized approximation scheme), elle est détaillée dans [HLMP04, DFH$^+$04]

Résultats

APMC est un modèle checker distribué, on peut donc faire tourner plusieurs machines en même temps via un cluster. Nous avons utilisé pour nos expériences un cluster de 75 Pentium



($\lambda = 96$, $\sigma = 3$, $\alpha = 6$)

FIG. 5.12 – Probabilité d'émission en fonction de la deadline.

IV 2 GHz avec 512 Mb de RAM. Nous avons considéré le paramètre d'approximation $\epsilon = 10^{-2}$, enfin nous avons pris les constantes réelles du protocole $\lambda = 780, \sigma = 24, \alpha = 10$.

Nous avons aussi, afin de vérifier nos hypothèses, fait les calculs pour les constantes prises sous PRISM et nous obtenons les mêmes résultats (ce qui prouve la cohérence des deux model-checkers) qui sont décrits dans le figures 5.12 et 5.13.

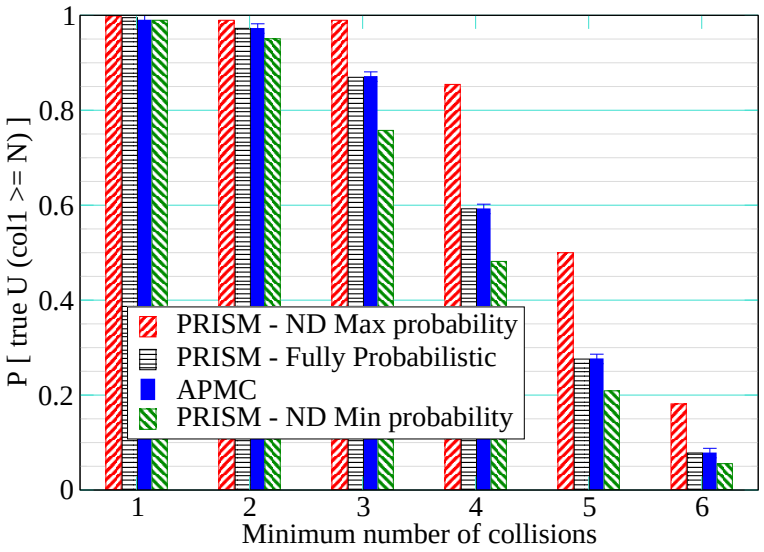
Enfin les résultats avec les constantes réelles sont donnés en figure 5.15 pour le nombre de collision et en figure 5.14 pour la probabilité d'envoi d'un message.

La figure 5.15 nous fait remarquer que tant que $2^N \leq \sigma$ la probabilité d'avoir une collision est environ 1, en effet, les deux stations ont de très grandes chances de tirer deux nombres aléatoires qui engendreront à nouveau une collision. Ensuite cette probabilité décroît assez rapidement. Le nombre de collisions engendrées par le protocole est assez faible ('a partir de 10 la probabilité est très faible).

La figure 5.14 comme celle 5.10 nous donne la probabilité pour qu'un message soit correctement envoyé par une des deux stations : on remarque à nouveau le premier seuil avant lequel la probabilité est nulle, et nous pouvons même donner une formule analytique de ce seuil :

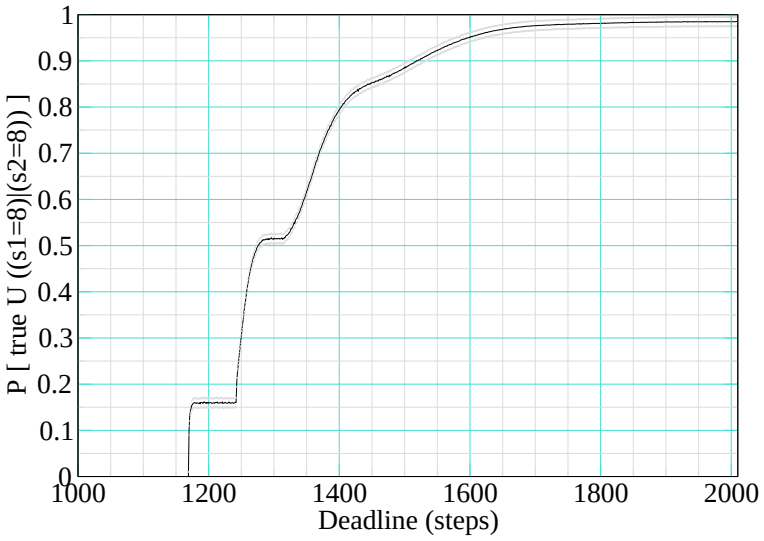
$$T_0 \simeq \overline{T}_{\text{Backoff}(N_{min})} + 3 \times N_{min} \times \sigma + \lambda$$

dans lequel N_{min} est le nombre de collisions inévitables, et Backoff(N_{min}) et le temps moyen passé dans les procédures de backoff correspondants à ces collisions. Pour les valeurs réelles du protocole nous avons $N_{min} = 5$ et Backoff(N_{min}) $= 31$.



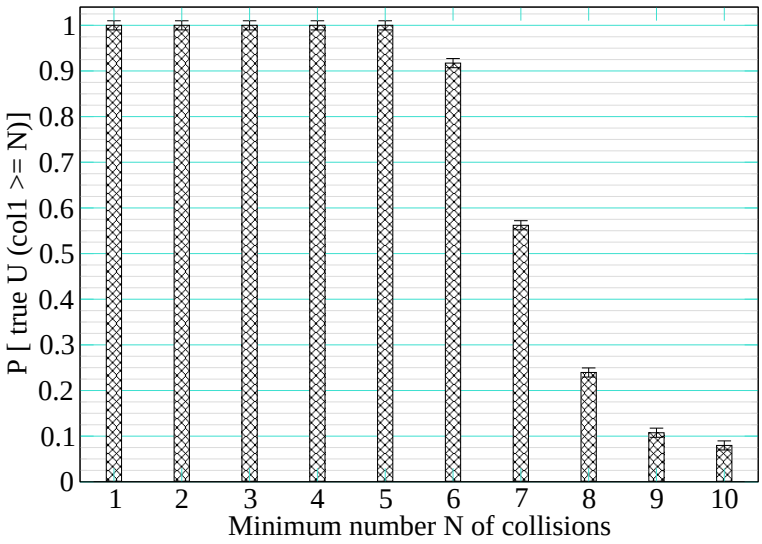
($\lambda = 96$, $\sigma = 3$, $\alpha = 6$, $deadline = 210$)

FIG. 5.13 – Probabilité d'avoir plus de N collisions, en fonction de N.



($\lambda = 780$, $\sigma = 24$, $\alpha = 10$)

FIG. 5.14 – Probabilité d'émission en fonction de la deadline.



($\lambda = 780$, $\sigma = 24$, $\alpha = 10$, $deadline = 2000$)

FIG. 5.15 – Probabilité d'avoir plus de N collisions, en fonction de N.

Après ce seuil la probabilité augmente rapidement et tend vers 1, par exemple après 2λ la probabilité d'envoyer correctement un message est de $0,9$.

Enfin on peut remarque l'émergence de plateaux. Ce ne sont pas des résultats surprenants (puisque déjà remarqués dans [GT88, WK99]). Ils sont dûs aux deadlines trop petites et ne permettant pas aux stations d'entrer dans la procédure de backoff.

La complémentarité des deux model-checkers nous a donc permis de vérifier, pour la première fois à notre connaissance, un grand nombre de propriétés quantitatives de CSMA/CD.

Troisième partie

Convergence des systèmes paramétrés probabilistes

Chapitre 6

Application des techniques de coupling

Cette chose plus compliquée et plus confondante que
l'harmonie des sphères : un couple.
Julien Gracq

La suite de la thèse sera consacrée à l'étude des propriétés de convergence des algorithmes probabilistes. Nous avons cherché à appliquer les techniques utilisées dans l'étude des systèmes probabilistes issues d'autres domaines scientifiques (Mathématiques, Physique statistique). Il se trouve que de cette manière, de nombreux outils peuvent nous aider à étudier les systèmes informatiques.
Dans ce chapitre nous nous intéresserons surtout au coupling, technique permettant de majorer le temps de convergence des algorithmes. Les notations que nous prendrons sont celles de [Ran03, FMP04c]

6.1 Motivations

La méthode de Monte-Carlo pour les chaînes de Markov est très efficace pour l'étude des systèmes sur des ensembles de très grande taille, l'idée de base est de simuler une marche aléatoire sur les configurations de l'ensemble en question. Ensuite, et même si les configurations n'ont que très peu de voisins, la chaîne de Markov engendrant la marche aléatoire va converger (vite ?) vers une distribution intéressante sur l'ensemble des configurations. Les fondations mathématiques de cette théorie peuvent se trouver [Bré99, JS96], nous ne donnerons ici que les résultats utiles dans le cadre de l'étude des systèmes informatiques distribués.
De nombreux domaines différents ont utilisé ces techniques durant ces quinze dernières années. Nous pouvons citer par exemple, l'estimation du volume d'un corps convexe [DFK92] et celle du permanent d'une matrice [JSV01]. Les versions exactes de ces deux problèmes sont des problèmes $\sharp$P-complets [JS96]. On peut aussi citer [Kar02] pour une vue d'ensemble de toutes les applications.
La raison la plus courante de l'intérêt porté aux techniques de majoration du temps de convergence d'une chaîne de Markov est celle de l'échantillonnage d'ensemble de très grande taille. En voici les principaux termes.

6.1.1 Echantillonnage

Les chaînes de Markov (voir toutes les définitions au chapitre 2) sont des outils puissants quand on a besoin d'échantillonner un ensemble Ω fini de configurations. L'idée est de connecter ensemble les éléments de Ω en choisissant pour chacun d'eux un nombre (de préférence petit) de voisins. Ensuite, partant d'un état arbitraire de Ω, la marche visitera les différentes configurations, se baladant de l'une à l'autre. Illustrons cela à l'aide d'un exemple simple.

Exemple 6.1. (Ensembles indépendants) *Imaginons que nous cherchons, par exemple, à échantillonner des ensembles indépendants (i.e. des sommets non reliés par des arêtes) d'un graphe fini. Une manière assez simple de connecter l'espace d'états (l'ensemble de tous les ensembles de sommets) est de n'autoriser les transitions entre deux configurations I et I' (ici des ensembles) que si leur distance de Hamming est $\phi(I, I') = 1$. La distance de Hamming étant définie par le nombre d'éléments qui ne sont que dans un des deux ensembles, on peut simuler une marche aléatoire $(I_t)_{t\in\mathbb{N}}$ de la manière suivante.*
Prendre I_0 l'état initial (par exemple l'ensemble vide).
Puis répéter :

1. *Choisir un sommet v suivant une distribution uniforme sur V (l'ensemble des sommets du graphe).*
2. *Puis*
   - *Si $v \in I_t$ on l'enlève : $I_{t+1} = I_t - \{v\}$*
   - *Si $v \notin I_t$ on le rajoute si c'est possible soit :*
     - *$I_{t+1} = I_t \cup \{v\}$ si v n'est voisin d'aucun des éléments de I_t*
     - *$I_{t+1} = I_t$ sinon.*

On peut ainsi décrire la chaîne de Markov sur l'ensemble des ensembles indépendants par ses probabilités de transitions :
- *$p(I, I') = 1/n$ si $\phi(I, I') = 1$*
- *$p(I, I') = 0$ si $\phi(I, I') > 1$*
- *$p(I, I') = 1 - \sum_{J\neq I} p(I, J)$ si $I = I'$*

La suite de résultats que nous allons présenter vont expliquer pourquoi cette chaîne est une bonne candidate pour l'échantillonnage. ♣

Définition 6.2. (Ergodicité) *Une chaîne de Markov est dite ergodique si elle est :*

1. *Irréductible, i.e. $\forall x, y \in \Omega$ il existe t tel que $p^t(x, y) > 0$.*
2. *Apériodique, i.e. $\forall x, y \in \Omega$ le PGCD de l'ensemble : $\{t | p^t(x, y) > 0\}$ vaut 1.*

On peut remarquer que la chaîne de Markov sur les ensembles indépendants est irréductible. En effet, on peut aller de n'importe quelle configurations à l'ensemble vide, en enlevant des sommets. De plus, cette chaîne de Markov est aussi apériodique. En effet la plupart des configurations ont des transitions sur elles-mêmes dues au fait que certains sommets ne peuvent être rajoutés.
On peut rendre facilement une chaîne de Markov apériodique en ajoutant des transitions sur elles-mêmes aux configurations. Cette opération ne change pas trop le temps de convergence des algorithmes.

Les deux lemmes suivants décrivent les principales propriétés des chaînes de Markov ergodiques, et nécessitent une définition préalable.

Définition 6.3. (Distributions stationnaires) *Etant donnée une chaîne de Markov de matrice de transition p, une distribution π sur Ω est dite stationnaire si elle satisfait*

$$^t\pi = {}^t\pi p,$$

soit, plus précisément, pour tout i de Ω :

$$\pi(i) = \sum_{j\in\Omega} \pi(j)p_{ij}$$

Cette équation est appelée équation du bilan globale.

Lemme 6.4. (Convergence vers la distribution stationnaire) *Une chaîne de Markov à espace d'états fini ergodique converge vers une unique distribution stationnaire π,i.e., pour tout $x, y \in \Omega$, on a :*

$$\lim_{t\to\infty} p^t(x, y) = \pi(y).$$

Lemme 6.5. (Equation du bilan détaillée) *Soit M une chaîne de Markov sur Ω ayant pour matrice de transition p. Si $\pi : \Omega \to [0, 1]$ est une fonction satisfaisant à l'équation du bilan dite* détaillée *:*

$$\forall x, y \in \Omega \qquad \pi(x)p(x, y) = \pi(y)p(y, x)$$

et si elle satisfait $\sum_{x\in\Omega} \pi(x) = 1$ alors elle est l'unique distribution stationnaire de M.

Les preuves de ces deux lemmes classiques se trouvent dans de nombreux ouvrages, parmi lesquels on peut citer [Bré99].

Exemple 6.6. *Dans notre exemple sur les ensembles indépendants il est évident que pour tous ensembles I, I' on a $p(I, I') = p(I', I)$, on conclut grâce au lemme 6.5 que l'unique distribution stationnaire de cette chaîne de Markov ergodique est la distribution uniforme.*
Une fois la stationnarité atteinte, l'objective d'échantillonnage sera lui aussi atteint, en effet tous les ensembles indépendants du graphe étudié auront la même probabilité d'apparaître.

♣

Il reste maintenant deux questions cruciales à résoudre. La première est comment faire pour échantillonner suivant des distributions plus complexes que la distribution uniforme, sera résolu dans le paragraphe suivant. La seconde est de savoir combien de temps il faut simuler la chaîne de Markov avant d'être assez proche de la distribution stationnaire. La résolution de ce problème et ses applications occupera une grande partie de la suite de la thèse.

6.1.2 Algorithme de Metropolis

L'algorithme de Metropolis Hastings [Mol02] est remarquablement simple. C'est le point de départ des techniques d'échantillonnage. Il nous permet de donner une technique systématique pour construire une chaîne de Markov qui converge vers une distribution fixée π.
Algorithme de Metropolis :
Soit un voisinage fixé entre les configurations.
Partant d'une configuration initiale x, on répète :

1. Choisir un voisin y de x uniformément et de manière aléatoire avec la probabilité $1/2\Delta$ où Δ est le nombre maximum de voisins.
2. Se rendre en y avec la probabilité $p = \min(1, \frac{\pi(y)}{\pi(x)})$.
3. Rester en x, avec la probabilité $1 - p$.

En utilisant le lemme 6.5 sur l'équation du bilan détaillée, on vérifie que π est bien la distribution stationnaire de cette chaîne de Markov. En effet, si x et y sont voisins (supposons sans perte de généralités $\pi(y) \leq \pi(x)$, alors $p(x,y) = \frac{1}{2\Delta}\frac{\pi(y)}{\pi(x)}$ et $p(y,x) = \frac{1}{2\Delta}$ et on a donc bien l'équation du bilan vérifiée (en effet si x et y ne sont pas voisins $p(x,y) = p(y,x) = 0$).

Exemple 6.7. *Reprenons notre exemple simple sur les ensembles indépendants, et supposons à présent que nous voulons échantillonner suivant la distribution non uniforme*

$$\pi(I) = \lambda^{|I|}/Z,$$

où $Z = \sum_{J\in\Omega} \lambda^{|J|}$ est une constante normalisante. Cette distribution correspond à une distribution naturelle sur les ensembles indépendants (voir par exemple [Ran03]). Comme auparavant, on définit les ensembles voisins par ceux qui ont une distance de Hamming inférieure à 1. Soit I et I' deux tels ensembles avec $I' = I \cup \{v\}$. Alors on a $\pi(I') = \lambda\pi(I)$.
L'algorithme de Metropolis nous enseigne donc, qu'afin d'échantillonner cet ensemble suivant cette distribution, il faut construire la chaîne de Markov de la manière suivante :

$$p(I, I') = \frac{1}{2N}\min(1, \lambda)$$

et,

$$p(I', I) = \frac{1}{2N}\min(1, \lambda^{-1})$$

On peut remarquer que la constante normalisatrice n'apparaît plus dans la description de la chaîne de Markov. Heureusement, d'ailleurs, car nous n'avons aucun moyen de la calculer.
♣

Il reste donc maintenant une seule question fondamentale, combien de temps devons nous simuler la chaîne pour être assez proche de la distribution stationnaire ?

6.2 Coupling

Cette partie du chapitre sera consacrée à la description d'une technique particulièrement utile pour la mesure du temps de convergence des chaînes de Markov ergodiques, celle du coupling.
Il nous faudra tout d'abord définir le temps de convergence (temps de mélange), puis décrire en détail les différentes étapes du coupling. L'ensemble de ces résultats peut se retrouver dans de nombreuses références (citons par exemple [Bré99]).

6.2.1 Temps de mélange

La notion classique permettant de mesurer le temps de convergence d'une chaîne de Markov ergodique vers sa distribution stationnaire est appelée le temps de mélange. Ce temps se mesure en terme de distance en variation entre l'état courant de la chaîne à l'instant t et la distribution stationnaire π.
Pour une comparaison entre les différents temps de convergence, on pourra étudier les références [AFar, LW98].

Définition 6.8. (Distance en variation) *Soit $p^t(x,y)$ la probabilité de se rendre de x à y en t étapes. La distance en variation totale au temps t est définie par :*

$$||p^t, \pi||_{vt} = \max_{x\in\Omega} \frac{1}{2} \sum_{y\in\Omega} |p^t(x,y) - \pi(y)|.$$

On notera $||p^t(x,.), \pi|| = \frac{1}{2}\sum_{y\in\Omega} |p^t(x,y) - \pi(y)|$.

Cela correspond à la norme L_1, le facteur $\frac{1}{2}$ étant introduit pour que la distance soit au plus 1. Nous pouvons à présent donner la définition suivante.

Définition 6.9. (Temps de mélange) *Pour $\epsilon > 0$, le temps de mélange $\tau(\epsilon)$ est défini par :*

$$\tau(\epsilon) = \min\{t : ||p^{t'}, \pi||_{vt} \leq \epsilon, \forall t' \geq t\}.$$

On dira qu'une chaîne de Markov est rapidement mixante si le temps de mélange est polynomial en N et en $log(\epsilon^{-1})$.

Remarques :

La grandeur classique permettant de mesurer la notion de convergence vers la distribution stationnaire est plutôt le rayon spectral. Ce rayon spectral est défini comme la valeur de la seconde plus grande valeur propre de la matrice de transition de la chaîne.
Nous pouvons remarquer tout d'abord que les deux notions sont des grandeurs voisines. En effet des résultats de [Sin93, AFar] relient les deux notions.

Le temps de mélange est une grandeur cruciale. Par exemple, pour l'étude du nombre de solutions de problèmes NP-complets (problèmes de comptage que l'on appelle ♯P-complets), la connaissance du temps de mélange permet d'estimer le nombre de telles solutions. En effet, dans la majorité des cas, il suffit de simuler une marche aléatoire sur l'ensemble des solutions, qui converge (rapidement) vers la distribution uniforme sur les solutions, pour obtenir une bonne approximation du nombre de solutions. Le calcul du temps de mélange se fait souvent par coupling car il est difficile de mesurer la distance vers une distribution stationnaire qu'on ne connait pas. On peut trouver plusieurs exemples d'application (coloriage de graphe, problème du sac à dos, recouvrement de graphes) et la théorie dans [JS96] par exemple.

De plus, il est souvent impossible (en informatique en particulier) de calculer les valeurs propres. En effet la taille des systèmes est souvent exponentielle en le nombre de machines et les matrices sont souvent impossibles à décrire.

Le coupling nous permet donc d'estimer cette notion. Sa simplicité et sa puissance l'ont rendu extrêmement populaire. En voici les notions principales.

6.2.2 Résultats

Définition 6.10. (Coupling) *Etant donnée une chaîne de Markov M, un coupling est une chaîne de Markov sur $\Omega \times \Omega$ définissant un processus stochastique $(X_t, Y_t)_{t\in\mathbb{N}}$ avec les propriétés suivantes :*

1. *Chacun des deux processus X_t et Y_t est une copie exacte de la chaîne de Markov d'origine M (Etats initiaux donnés $X_0 = x$ et $Y_0 = y$).*
2. *Si $X_t = Y_t$, alors $X_{t+1} = Y_{t+1}$,*

La condition 1 assure que chaque processus vu isolément simule la chaîne de Markov originale, mais le coupling fait évoluer les deux chaînes simultanément et de manière à les rendre égales, après quoi la condition 2 assure qu'elles ne se sépareront plus.
La notion de temps de coupling, temps avant lequel les deux processus tombent d'accord sera (on le démontrera) une bonne borne pour le temps de mélange.

Définition 6.11. (Temps moyen de coupling) *Pour des états initiaux x et y on notera :*

$$T^{x,y} = \min\{t | X_t = Y_t, X_0 = x, Y_0 = y\}$$

et nous définissons le temps de coupling par :

$$T = \max_{x,y} E(T^{x,y})$$

Le théorème suivant relie le temps de coupling et le temps de mélange :

Théorème 6.12. $\forall \epsilon > 0$

$$\tau(\epsilon) \leq \lceil Telog(\epsilon^{-1}) \rceil.$$

Démonstration. La preuve du théorème 6.12 repose sur deux lemmes connus en théorie des probabilité : le lemme de coupling et l'inégalité de Markov. En voici les énoncés ; les preuves peuvent se trouver dans [Bré99, AFar].

Lemme 6.13. (Lemme de coupling) *Soit deux variables aléatoires X et Y sur Ω avec des distributions respectives μ et ν, on a :*

$$||\mu - \nu|| \leq \mathbb{P}[X \neq Y].$$

Lemme 6.14. (Inégalité de Markov) *Soit une variable aléatoire réelle X et soit $a > 0$*

$$\mathbb{P}(X > a) \leq \frac{E(X)}{a}$$

Pour appliquer ces résultats dans le contexte de notre chaîne de Markov, considérons que Y_t est le processus stationnaire, i.e., si on choisit Y_0 suivant la distribution stationnaire π, tous les Y_t seront aussi distribués suivant π. Si nous appliquons donc le lemme de coupling aux variables aléatoires X_t et Y_t on a :

$$||p^t(x, .), \pi|| \leq \mathbb{P}[X_t \neq Y_t] \leq \max_{x,y} \mathbb{P}[T^{x,y} > t].$$

Ainsi, si on note $T_c = \min_{x,y}\{t|\mathbb{P}[T^{x,y} > t] \leq e^{-1}\}$ (grandeur appelée temps de coupling) on a

$$\mathbb{P}[T^{x,y} > kT_c] \leq e^{-k}$$

pour tout k entier et tout x, y dans Ω. Pour prouver cela, il suffit de considérer k périodes de longueur T_c ; à chaque époque le coupling échoue avec probabilité au plus e^{-1}, par définition de T_c.
On peut maintenant appliquer les deux inégalités ; on obtient

$$\tau(\epsilon) \leq T_c\lceil log(\epsilon^{-1})\rceil.$$

On applique maintenant l'inégalité de Markov (lemme 6.14) ; on obtient de manière évidente que

$$\tau(\epsilon) \leq Te\lceil log(\epsilon^{-1})\rceil.$$

□

Intérêt des preuves par coupling :

Les preuves par coupling permettent d'obtenir efficacement des bornes pour le temps de mélange. Cette technique est très utile car la recherche du temps de coupling ne requiert pas une connaissance de la distribution stationnaire. Or, il existe de nombreux algorithmes (détaillés dans [Sin97]) souvent NP-complets pour lesquels la distribution stationnaire n'est pas connue.

Prenons par exemple, le problème bien connu dit du sac à dos. Rappelons que ce problème consiste à trouver quelles sont les manières de remplir un sac de volume b avec des objets de volume $a_1, a_2, ...a_n$. Afin de déterminer le nombre de solutions de ce problème on construit une marche aléatoire sur l'ensemble des solutions et on essaye de savoir le moment où elle se rapproche de la distribution stationnaire (uniforme sur les solutions). Les techniques de coupling permettent dans ce cas où la distribution stationnaire n'est absolument pas connue de calculer ce temps de manière efficace et simple (voir [Sin97]).
Afin d'illustrer la technique nous pouvons l'appliquer à un autre exemple très simple :

Exemple 6.15. (Marche aléatoire sur l'hypercube). *Nous considérons le problème simple de choisir au hasard un sommet sur un hypercube de dimension N. Nous avons donc $\Omega = \{0,1\}^N$. Une marche aléatoire simple se balade sur les sommets de l'hypercube, en changeant la valeur de une des N coordonnées choisie au hasard. Afin de pouvoir appliquer les résultats de coupling, on change un peu la marche aléatoire en autorisant à rester sur le même sommet. Voici donc la description de la chaîne de MarkovM_{cube} correspondante :*

On part du sommet $X_0 = (0, 0, ..., 0)$, puis on répète :

1. *On choisit $(i, b) \in \{1, ..., N\} \times \{0, 1\}$ aléatoirement et de manière uniforme.*
2. *On passe de X_t à X_{t+1} en mettant la valeur du i^{me} à b.*

Si on appelle ϕ la distance de Hamming (nombre de sites qui diffèrent), la matrice de transition de la chaîne de Markov M_{cube} peut être définie par :

- $p(X,Y) = 1/2N$ *si* $\phi(X,Y) = 1$
- $p(X,Y) = 1/2$ *si* $X = Y$
- $p(X,Y) = 0$ *sinon.*

Il est facile de prouver que cette chaîne de Markov est ergodique et vérifie l'équation du bilan détaillée pour la distribution uniforme.
Nous définissons alors le coupling le plus simple qui soit, en choisissant le même site i *et la même valeur* b *pour les deux processus* X *et* Y*.*
Cela définit un coupling : chacun des deux processus se comporte comme la chaîne de Markov initiale et si les deux processus tombent d'accord, ils le restent, puisqu'ils feront exactement la même chose.
Nous cherchons donc le temps de coupling, on notera D_t *la variable aléatoire* $\phi(X_t, Y_t)$*. Le but est de trouver le temps moyen avant lequel* $D_t = 0$ *sachant que* $D_0 = N$ *; en effet lorsque* $D_t = 0$ *les deux processus sont couplés.*
Supposons qu'à l'instant t*,* $D_t = d$*. Si on tire au hasard un des* d *sites qui diffèrent, alors* $D_{t+1} = D_t - 1$ *sinon il ne change pas, on a donc :*

- $D_{t+1} = d - 1$ *avec probabilité* $\frac{d}{N}$
- $D_{t+1} = d$ *avec probabilité* $1 - \frac{d}{N}$

Ainsi, pour des configurations initiales x *et* y *le temps* $T^{x,y}$ *est majoré par la variable aléatoire* $T_N + T_{N-1} + ... + T_1$*, où* T_d *désigne le temps qu'il faut pour que* D_t *passe de* d *à* $d-1$*. Ainsi* $E(T_d) = N/d$ *(espérance du temps avant lequel une pièce biaisée tombe sur face). On obtient*

$$E(T) = O(N \ln N)$$

(quand $N \to \infty$*). En appliquant le théorème 6.12, on obtient le temps de mélange de la chaîne* M_{cube} *:*

$$\tau(\epsilon) = O(n \ln(n\epsilon^{-1})).$$

♣

La fin de ce chapitre sera consacrée à l'étude (que nous avons menée dans [FMP04c]) d'un algorithme très largement utilisé dans plusieurs domaines scientifiques : l'asymmetric exclusion problem (ASEP). Nous le décrirons dans la section 6.4. En guise de préliminaires, on doit rappeler plusieurs résultats concernant un problème connu de théorie des probabilités : la ruine du joueur.

6.3 Ruine du joueur

La ruine du joueur est un des premiers problèmes que l'on étudie en théorie des probabilités. En effet, sa formulation est très simple et ses applications nombreuses.
Informellement il s'agit d'un joueur (au casino par exemple) qui a une fortune initiale et un objectif de gain ; il joue à un jeu dont les probabilités de gain et de perte sont connues. Le but du problème est de savoir son espérance de gain et le temps d'arrêt du jeu.
En voici des termes plus précis. Les résultats que nous donnons dans les sections suivantes peuvent se retrouver dans de nombreux ouvrages dont [Fel70]. Nous ne reprendrons pas les preuves ici.

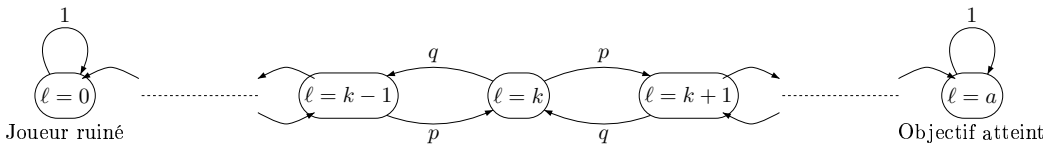


FIG. 6.1 – L'évolution de la variable aléatoire X_t pour un jeu de ruine du joueur

6.3.1 Problème

Considérons un joueur qui gagne un euro avec probabilité p et qui en perd un avec probabilité q en une partie. Sa fortune (capital) initiale est de z. Il joue contre un joueur de fortune supposée infinie, et a pour objectif de gain un montant a. Le jeu s'arrête donc lorsque la fortune du joueur (on la notera $X_t = \ell$ après t parties) est nulle (il est ruiné) ou est de a (il a atteint son objectif).
Le problème classique de la ruine du joueur s'intéresse au calcul de grandeurs, la probabilité de ruine et la temps moyen d'un jeu (i.e., le nombre de parties avant que l'on s'arrête).
Dans le problème classique $p = q = \frac{1}{2}$, mais nous verrons par la suite que ce n'est pas indispensable pour calculer les grandeurs qui nous intéressent. La description graphique de l'évolution de la fortune du joueur est donnée en figure 6.1

Comme on peut le remarquer ce problème est un problème de marche aléatoire classique avec deux barrières absorbantes (en 0 et a). Ces problèmes ont de nombreuses applications notamment en physique dans le cas de la diffusion de particules suivant un axe. Les physiciens utilisent ce modèle pour le mouvement brownien sur un axe sur lequel une particule est soumise à un grand nombre de chocs qui déterminent son mouvement aléatoire. Le cas classique $p = q = 1/2$ représente le cas où il y a autant de chocs par la droite que par la gauche (jeu équilibré). Le cas $p > q$ (cas déséquilibré et favorable pour le joueur) représente, lui, le cas où les chocs venus de la gauche sont plus probables.
Par la suite nous exploiterons des résultats dans le cas équilibré $p = q$, mais dans la section suivante nous donnerons les théorèmes dans le cas général.

6.3.2 Résultats

Rappelons que l'ensemble des résultats décrits ici, peuvent se retrouver dans [Fel70] entre autres.

Le premier résultat nous donne la probabilité de ruine ou de gain d'un joueur.

Théorème 6.16. (Probabilité de ruine) *Soit un joueur avec une fortune initiale z et un objectif de gain a (le gain net est en fait $a - z$). Alors la probabilité pour qu'il se ruine est q_z et celle pour qu'il atteigne son objectif est $1 - q_z$ où*

$$q_z = \frac{(q/p)^a - (q/p)^z}{(q/p)^a - 1}.$$

si $p \neq q$ et vaut

$$1 - \frac{z}{a}$$

sinon. Le gain du joueur (G) vaut donc soit $-z$ soit $a-z$. Donc le gain moyen est de $E(G) = a(1-q_z) - z$. Notons que le gain moyen est nul dans le cas d'un jeu équilibré.

Le deuxième résultat porte sur le nombre de parties nécessaires avant qu'un jeu ne s'arrête.

Théorème 6.17. (Durée moyenne d'un jeu) *Soit un jeu de ruine du joueur, avec pour fortune initial z et pour objectif de gain a. Alors la durée moyenne d'un jeu vaut D_z où*

$$D_z = \frac{z}{q-p} - (\frac{a}{q-p})(\frac{1-(q/p)^z}{1-(q/p)^a})$$

si $q \neq p$ et vaut

$$D_z = z(a-z)$$

sinon.

On peut remarquer que cette durée est beaucoup plus longue que celle qu'on pouvait intuiter. En effet, imaginons un joueur muni d'une fortune initiale de 500 euros, et jouant à un jeu équilibré (objectif de gain 1000 euros), il lui faudra 250000 parties en moyenne avant de gagner ou d'être ruiné.

6.4 Etude de cas : ASEP pour 2 particules

L'asymmetric simple exclusion process (ASEP) a attiré l'intérêt du fait des nombreuses applications dans différents domaines, comme par exemple la gestion des embouteillages, la cinétique de la biopolymérisation ou le flux de particules. Mais, plus généralement, parce qu'il est un modèle paradigmatique pour les systèmes non équilibrés (voir [DEHP93]). Nous avons considéré dans [FMP04c] une version discrète de l'ASEP sur une topologie en anneau, car cette version se rapproche le plus des problématiques que nous nous posons en informatique distribuée. Enfin les mises à jours sont séquentielles et faites dans le sens inverse des aiguilles d'une montre (voir figure 6.2).

Le modèle peut être défini comme suit. On considère N sites, numérotés (dans le sens des aiguilles d'une montre) de 1 à N sur un anneau (le site 1 est immédiatement à droite de N). Chaque site i peut être occupé par une particule ($x_i = 1$) ou vide ($x_i = 0$). Une configuration $X = (x_1, ..., x_N)$ est un uplet de $\Omega = \{0,1\}^N$. Soit une configuration X. S'il y a une particule au site N et qu'il n'y en a pas au site 1 alors la particule avance vers la droite (i.e., elle se rend au site 1) avec probabilité p. Sinon elle reste sur le site $i = N$. Nous continuons à mettre à jour en prenant $i - N - 1$ puis nous mettons tous les sites a jour jusque $i = 1$. Un tour est alors effectué. On définit alors une chaîne de Markov $M(N, p, m)$ que nous étudierons dans la partie 6.4.1 où N est la taille de l'anneau, p la probabilité de saut d'un site à l'autre et m le nombre de particules présentes sur l'anneau.
Comme le dit Schutz dans [Sch97], les résultats exacts sur la dynamique de l'ASEP sont rares, même dans le cas d'un petit nombre de particules ($m = 2$, par exemple). Même si le système avec deux particules peut sembler trivial et sans intérêt, il a permis d'exhiber un grand nombre de propriétés dans le comportement de l'ASEP en densité finie, comme l'apparition

de la superdiffusion (voir [DEHP93]).
Ainsi, il nous a paru intéressant d'appliquer les techniques de coupling à cet algorithme, car sans être trivial il permet de mettre en lumière des techniques de preuves intéressantes.
Le calcul du temps de convergence de l'ASEP se trouve notamment dans [GS92]. Les résultats que nous obtenons par nos méthodes (dans [FMP04c]) sont identiques.

Autres travaux connexes :

Citons quelques travaux sur le sujet. Schutz étudie les aspects quantitatifs de l'ASEP avec deux particules mais dans le cas continu, sur une droite infinie [Sch97]. Cette étude a été adaptée sur un anneau par Priezzhev [Pri03], mais ces deux auteurs ne s'intéressent pas au temps de mélange.
Le temps de mélange dans le cas de l'ASEP a été traité dans [Fil91] et [CP00]. Mais, ces deux articles traitent, en fait, le cas d'un nombre arbitraire de particules, mais dans un cadre différent (totally asymmetric exclusion process) avec une mise à jour aléatoire et le saut d'un site à un autre non probabiliste ($p = 1$).
Nous avons donc choisi d'étudier le cas séquentiel sur un anneau. Nos méthodes d'analyse sont originales car basées sur le coupling.

6.4.1 ASEP vu comme une chaîne Markov

L'ASEP sur un anneau de taille N avec mises à jour séquentielles peut être étudié comme une chaîne de Markov $(X^t)_{t\in\mathbb{N}}$ définie sur l'espace $\Omega = \{0,1\}^N$, pour lequel $X^{t+1} = (x_1^{t+1}, ..., x_N^{t+1})$ est calculé à partir de $X^t = (x_1^t, ..., x_N^t)$ de la manière suivante :
Pour i variant de N à 1 faire :

$$(x_i^{t+1}, x_{i+1}^{t+1}) = \begin{cases} \begin{cases} (0,1) & \text{avec probabilité } p \\ (1,0) & \text{avec probabilité } q = 1-p \end{cases} & \text{si } (x_i^t, x_{i+1}^t) = (1,0) \\ (x_i^t, x_{i+1}^t) & \text{sinon} \end{cases}$$

Ici, t représente donc le nombre de tours effectués depuis le début du processus. Chaque tour est composé de N étapes élémentaires consistant en la mise à jour de tous les couples de sites contigus.
L'évolution d'une configuration X avec deux particules non contiguës est représentée dans la figure 6.2 : les mouvements des particules sont indépendants.

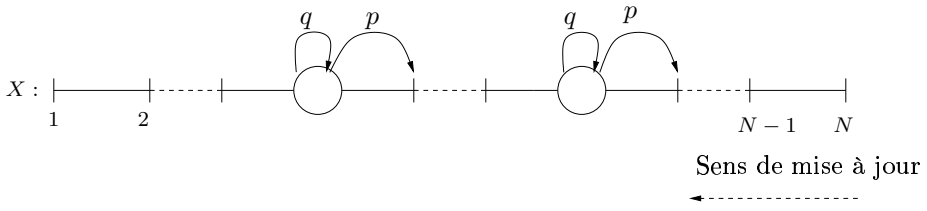


FIG. 6.2 – Evolution pendant un tour avec deux particules non contiguës.

L'évolution après un tour de X lorsque les deux particules sont contiguës est donnée dans la figure 6.3. On peut remarquer qu'il n'y a que trois mouvements possibles, en effet la parti-

cule de gauche est bloquée si celle de droite n'avance pas.

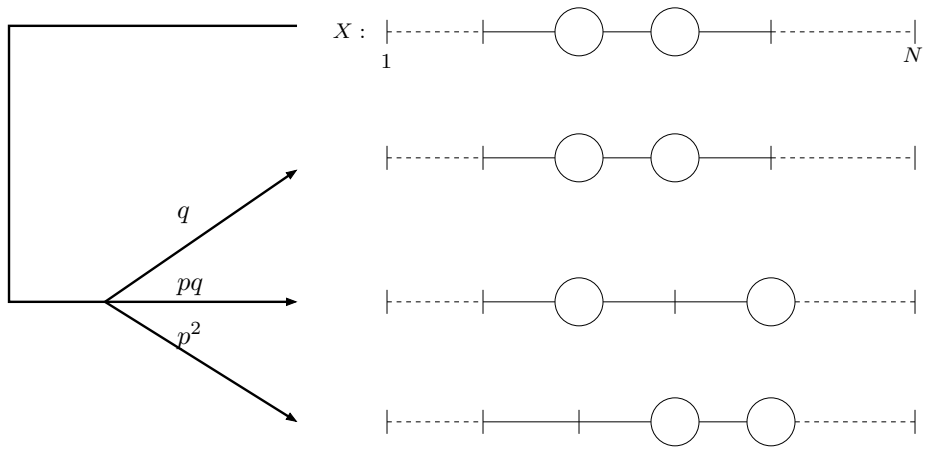


FIG. 6.3 – Evolution dans le cas de deux particules contiguës.

A présent nous pouvons vérifier que cette chaîne de Markov est ergodique et trouver sa distribution stationnaire.
La chaîne est irréductible toutes les configurations sont, en effet, accessibles les unes à partir des autres, il suffit de déplacer les particules comme on le désire. De plus, elle est apériodique car la probabilité de rester dans la même configuration après un tour est toujours strictement positive. Cette chaîne de Markov est donc ergodique. De plus, on remarque que la distribution uniforme vérifie l'équation du bilan globale, et donc c'est bien la distribution stationnaire.
Il faut donc à présent trouver un coupling afin de calculer le temps de mélange vers la distribution uniforme de cette chaîne de Markov.

6.4.2 ASEP vu comme un problème itéré de ruine du joueur

Coupling

Notre but est de trouver un coupling approprié et de calculer le temps de coupling. Nous dirons qu'une particule de $X = (x_1, ..., x_N)$ et $Y = (y_1, ..., y_N)$ coïncident si elles sont situées à la même position (i.e., $x_i = y_i = 1$).
Il faut donc trouver un coupling (X^t, Y^t) dans lequel, partant d'une configuration arbitraire (X_0, Y_0) où aucune particule ne coïncide, on se retrouve enfin à une configuration finale (X_f, Y_f) où les deux particules coïncident. Ce coupling se passe en deux phases :

Première Phase :
Pendant une première phase aucune particule ne coïncide (voir figure 6.4).

Après au plus N^2/pq tours en moyenne, il y a deux particules qui coïncident entre X et Y. En effet le problème du calcul de la durée de la première phase peut se résoudre grâce à une marche aléatoire avec deux sommets absorbants. Les résultats sur cette marche seront donnés dans la section suivante. Il suffit de considérer qu'il n'y a qu'une seule particule sur chaque anneau (quand deux particules sont contigus, alors avec probabilité p on passe de celle

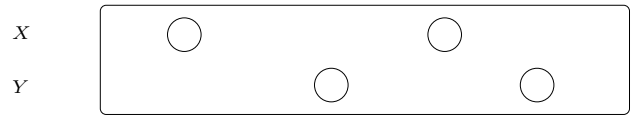


FIG. 6.4 – Un couple de configurations pendant la phase 1.

de gauche à celle de droite) et de trouver le temps avant qu'elles ne se rencontrent. Pour cela il suffit de considérer la distance (en sites) entre la particule de X et celle de Y, pour obtenir la marche aléatoire.

Seconde phase :
La seconde phase commence alors. Pendant cette période on va passer d'un couple de particules coïncidentes à deux couples de particules coïncidentes.
On suppose donc qu'une particule de X coïncide avec une particule de Y, dans toutes les figures suivantes, ces particules seront représentés en noir. La particule noire de X sera notée b_X (resp. b_y pour celle de Y), parfois b. Nous supposerons que la particule blanche de X est située à gauche de b_X (notion toute relative vu que l'on étudie le processus pour un anneau, mais utile pour les figures et pour la compréhension), et celle de Y à droite de b_Y.
Soit ℓ la distance (de gauche à droite) de la particule noire b à la particule blanche de droite, w_y.
Soit d la distance de la particule de gauche, w_y, à celle de droite, w_y. Tout ceci est décrit dans la figure 6.5

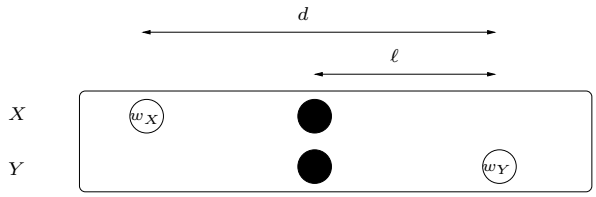


FIG. 6.5 – Un couple de configurations en phase 2.

Notre coupling consiste donc en phase 2 à :
- Faire avancer les particules blanches ensemble si c'est possible (lorsque b n'est pas immédiatement à la droite de w_X)
- Faire avancer les particules noires ensemble lorsque c'est possible (lorsque w_y n'est pas immédiatement à la droite de b)

Voici une explication détaillée de ce coupling :

- **Cas 1 :** b est isolé ; alors dans ce cas les particules noires avancent ensemble et les particules blanches aussi (voir figure 6.6).
- **Cas 2 :** b est juste à la droite de w_X ($\ell = d - 1; \ell > 1$). Les particules noires avancent ensemble. La particule w_Y avance indépendamment des noires. Il existe un sous-cas dans lequel w_X reste à la même position alors que w_Y avance ; dans ce cas $d' = d + 1$, voir la figure 6.7.

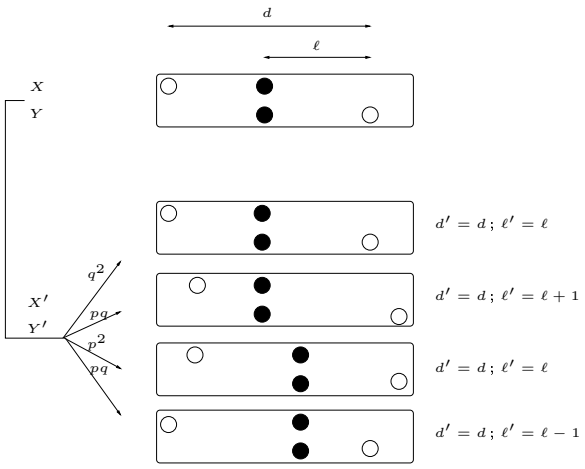


FIG. 6.6 – Evolution typique d'un couple

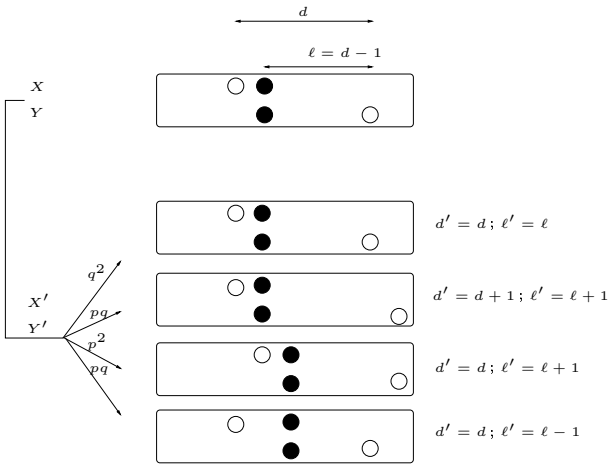


FIG. 6.7 – Evolution lorsque b et w_X sont contigus.

– **Cas 3 :** w_Y est juste à la droite de b ($\ell = 1, l < d - 1$). Dans ce cas, les particules blanches avancent ensemble. La particule noire b_X avance indépendamment. Il existe un seul sous-cas, dans lequel b_Y n'avance pas alors que b_X avance (voir figure 6.8). Dans ce sous-cas on est obligé de recolorer les particules car comme on peut le voir sur la figure 6.8, les particules qui coïncident ne sont plus les mêmes ; on passe donc à une configuration dans laquelle $d' = N - d + 1$.

Il y a, en plus un cas limite, dans lequel $\ell = 1$ et $d = \ell + 1 = 2$. En fait, ce n'est qu'une simple combinaison des cas 1 et 2.

Il est évident que ce coupling vérifie les conditions de définition d'un coupling, nous expliquons dans la partie suivante pourquoi ce coupling peut être vu comme un jeu de ruine du joueur, où d est l'objectif de gain et ℓ la fortune courante du joueur.

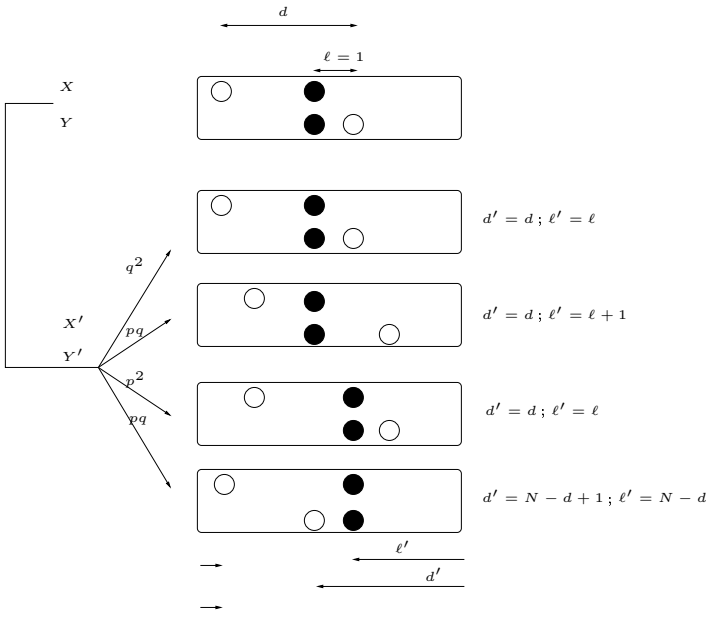


FIG. 6.8 – Evolution lorsque b et w_Y sont contigus.

Modélisation

Pendant un jeu, d reste généralement constant. Il y a pourtant deux cas limites correspondant au cas où d n'est plus préservé.

- **Cas limite 1 : le joueur a atteint l'objectif** ($\ell' = d$) ; b est juste à la droite de w_X ($\ell = d - 1$) ; dans ce cas w_X reste à la même position tandis que w_Y avance avec la probabilité p, ce qui entraîne $\ell' = d$. Ceci est décrit dans la figure 6.9. Un nouveau jeu commence avec $\ell := d; d := d + 1$.

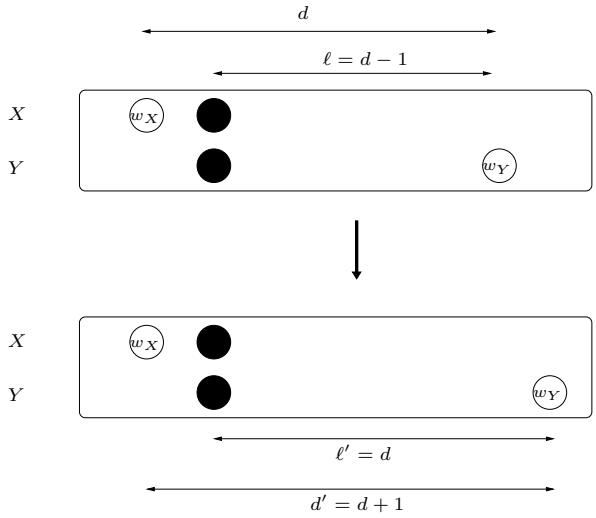


FIG. 6.9 – Cas limite 1.

– **Cas limite 2 : le joueur est ruiné** $\ell' = 0$: Ce cas se produit lorsque w_Y est juste à la droite de b_Y ($\ell = 1$). Dans ce cas b_Y reste à la même place tandis que b_X peut avancer avec probabilité p ce qui conduit à $\ell' = 0$. Après changement de couleur entre b_Y et w_Y un nouveau jeu commence avec $\ell := N - d; d := N - d + 1$; tout ceci est décrit dans la figure 6.10.

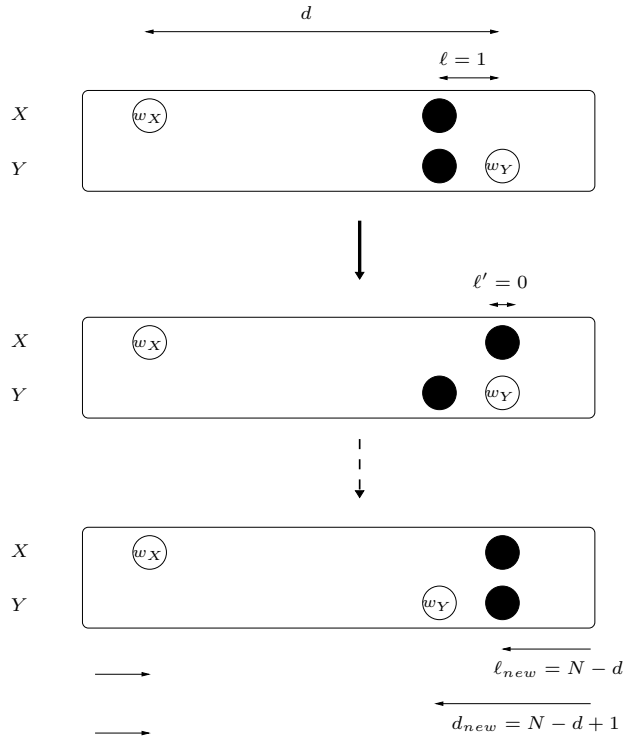


FIG. 6.10 – Cas limite 2.

Ainsi, l'évolution de ℓ peut être vue comme l'évolution de la fortune d'un joueur qui évolue à chaque tour de $+1$ (resp. 0, -1) avec probabilité pq (resp $p^2 + q^2$, pq) jusqu'à l'atteinte d'un des deux cas limites décrits précédemment. Ceci est décrit dans la figure 6.11.

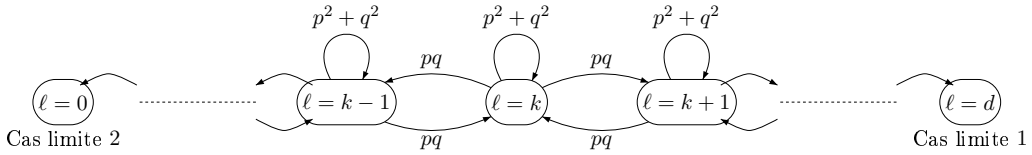


FIG. 6.11 – Un jeu de ruine du joueur.

A la fin d'un jeu, on redémarre un nouveau jeu avec des nouvelles valeurs pour d et pour l. Si on est arrivé au cas limite 1 on redémarre avec $d_{new} = d$ et $\ell_{new} = \ell + 1$; si on est arrivé au cas limite 2, on redémarre avec $d_{new} = N - d + 1$ et $\ell_{new} = N - d$.
Le dernier jeu est celui qui commence avec un objectif $d = N - 1$ et s'arrête dans le cas limite

1. Dans ce cas d augmente et devient égal à N ce qui implique que les deux particules blanches coïncident.
On peut écrire formellement la seconde phase de notre coupling comme un jeu itéré de ruine du joueur. En effet, si on note $jeu(d,\ell)$, le processus de jouer un jeu de ruine du joueur avec les probabilités données dans la figure 6.11, on peut itérer la boucle suivante :

FAIRE
Jouer le $jeu(d,\ell)$

1. Si le jeu s'arrête avec $\ell = d$ et $d < N-1$, alors :
   $D := d+1$; $\ell := d-1$, on recommence un nouveau jeu
2. Si le jeu s'arrête en $\ell = 0$, alors :
   $d := N-d+1$; $\ell := d-1$, on recommence un nouveau jeu
3. Si le jeu s'arrête avec $\ell = d$ et $d = N-1$, alors on arrête la boucle

On peut noter que chaque jeu (à part le premier) débute avec $\ell = d-1$. Nous pouvons alors donner le résultat suivant :

Proposition 6.18. *Soit d un entier fixé, on a pour le $jeu(d,d-1)$ les assertions suivantes :*

1. *La probabilité de terminer en $\ell = d$ (cas 1) est $\frac{d-1}{d}$*
2. *La probabilité de terminer en $\ell = 0$ (cas 2) est $\frac{1}{d}$*
3. *La durée moyenne d'un jeu est majorée par $\frac{d-1}{2pq}$ ($\leq \frac{N}{2pq}$)*

Démonstration. Ce jeu de ruine du joueur est un jeu équilibré mais dont la somme de perte et de gain ne font pas 1, de ce fait les deux premières assertions sont des conséquences immédiates du théorème 6.16. En effet, le fait que le jeu soit équilibré suffit pour déterminer les probabilités de sortir d'un côté ou de l'autre de la chaîne.
Par contre pour la troisième assertion il faut trouver le temps moyen que l'on reste dans chaque état sans y sortir. Notons T_a ce temps, alors on a $\mathbb{P}(T_a = k) = \alpha^k$ où $\alpha = p^2+q^2$. De ce fait :

$$E(T_a) = \sum_{k\geq 1} \alpha^k = \frac{\alpha}{1-\alpha} = \frac{1}{2pq} - 1 \leq \frac{1}{2pq}.$$

On multiplie donc le temps de sortie dans le cas de la ruine du joueur classique (voir le théorème 6.17 ici $a = d$ et $z = d-1$) avec le temps moyen d'attente dans chaque état pour obtenir la troisième assertion.

□

On peut donc représenter l'évolution de d entre deux jeux consécutif dans la figure 6.12.

L'évolution de d durant le jeu itéré est donnée en figure 6.13.

6.4.3 Résultats

La seconde phase du coupling (le couplage des deux dernières particules) prend en moyenne $\sum_{i=1}^{M} E(T_i)$, dans lequel M est une variable aléatoire correspondant au nombre de jeux, et T_i

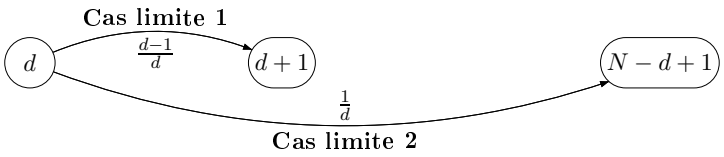


FIG. 6.12 – Evolution de d à la fin de chaque jeu.

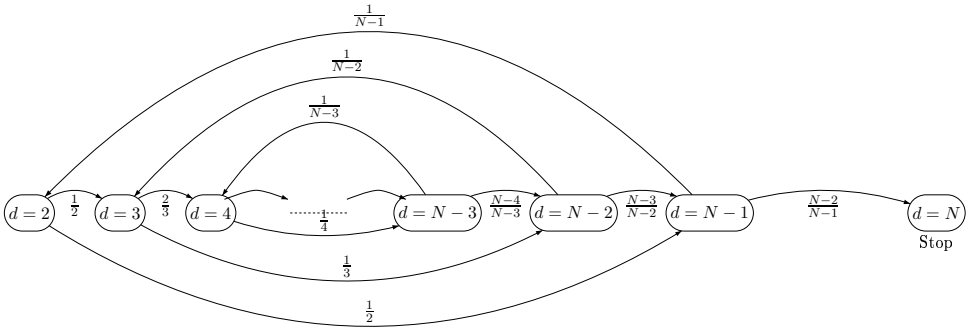


FIG. 6.13 – Evolution de d pendant le jeu itéré.

la durée du i^{me} jeu. En utilisant le lemme de Wald (voir par exemple [Bré99]), on a :

$$\sum_{i=1}^{M} E(T_i) \leq E(M) \times \max_i(E(T_i)) = E(M) \times \frac{N}{2pq}$$

On obtient une borne supérieure de M grâce à la proposition 6.20 donnée un peu plus loin, et elle vaut $2N$. On en déduit que la seconde phase prend en moyenne au plus $\frac{N^2}{pq}$ tours. De ce fait le temps de coupling du processus entier (incluant la première phase) est d'au plus $\frac{2N^2}{pq}$. On peut donc formuler le résultat suivant :

Théorème 6.19. *Le temps de mélange de l'ASEP sur un anneau avec deux particules est au plus* $\frac{2eN^2}{pq}\ln(\epsilon^{-1})$

Pour achever la preuve du théorème il nous reste donc à trouver une borne supérieure du nombre moyen de jeux, soit à trouver une borne supérieure pour l'absorption (état $d = N$) de la chaîne de Markov décrite en figure 6.13.

Considérons la matrice $(N-2)\times(N-2)$ $A = (a_{i,j})$ correspondante à la chaîne de Markov réduite aux états transitoires (on a $2 \leq i \leq N-1$ et $2 \leq j \leq N-1$). On a :

- $a_{k,k+1} = \frac{k-1}{k}$ pour $k = 2, ..., N-2$,
- $a_{k,N-k+1} = \frac{1}{k}$ pour $k = 2, ..., N-1$,
- $a_{i,j} = 0$, sinon.

On peut remarquer que, pour N pair et $k = \frac{N}{2}$: $a_{k,k+1} = a_{k,N-k+1} = 1$. La matrice A est la suivante :

$$A = \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 & \dots & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{2}{3} & 0 & \dots & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{3}{4} & \dots & 0 & \frac{1}{4} & 0 & 0 \\ \vdots & & & \ddots & \ddots & & & & \vdots \\ \vdots & & & & & \ddots & & & \vdots \\ \vdots & & & & & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \frac{1}{N-3} & 0 & \dots & 0 & 0 & \frac{N-4}{N-3} & 0 \\ 0 & \frac{1}{N-2} & 0 & 0 & \dots & 0 & 0 & 0 & \frac{N-3}{N-2} \\ \frac{1}{N-1} & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Nous devons donc calculer $\max_{2\leq i\leq N-1}(E_i)$ où E_i est le temps moyen pour la marche aléatoire décrite par la matrice A de se rendre dans l'état absorbant N depuis l'état i.

Proposition 6.20.

$$\forall i \in 2, ..., N-1, E_i \leq 2N.$$

Démonstration. La preuve est faite en trois étapes :

1. Calcul de $(E_i)_{2\leqslant i\leqslant N-1}$. Le vecteur $(E_i)_{2\leqslant i\leqslant N-1}$ est calculé par $(I-A)^{-1}\mathbf{1}$, où I est la matrice identité et $\mathbf{1}$ représente le vecteur dont toutes les coordonnées valent 1 (voir [Bré99]). Soit $B = (b_{i,j})$ défini par :

- $b_{i,j} = \frac{j}{2(N-j)}$ si $2 \leqslant j < i \leqslant N-j+1 \leqslant N-1$,
- $b_{i,j} = \frac{(i-1)(2N-i)}{2(j-1)(N-j)}$ si $2 \leqslant i \leqslant j \leqslant N-i+1 \leqslant N-1$,
- $b_{i,j} = \frac{(N+j-1)}{2(j-1)}$ si $1 \leqslant N-j+1 \leqslant i \leqslant j \leqslant N-1$,
- $b_{i,j} = \frac{(N-i)(N-i+1)}{2(j-1)(N-j)}$ si $N-i+1 \leqslant j < i \leqslant N-1$.

On peut vérifier que $B = (I-A)^{-1}$. Ainsi $(E_i)_{2 \leqslant i \leqslant N-1} = B\mathbf{1}$, donc :
$E_i = \sum_{j=2}^{N-1} b_{i,j}$, $\forall i \in \{1, ..., N-1\}$,

$$E_i = N \sum_{j=N-i+1}^{N-2} \frac{1}{j} + \frac{(i-1)(2N-i)}{N-1} \sum_{j=i-1}^{N-i} \frac{1}{j}, \quad \text{if} \quad 2 \leqslant i \leqslant \frac{N+1}{2},$$

$$E_i = N \sum_{j=i-1}^{N-2} \frac{1}{j} + \frac{(N-i)(N-i+1)}{N-1} \sum_{j=N-i+1}^{i-2} \frac{1}{j}, \quad \text{if} \quad \frac{N+1}{2} < i \leqslant N-1.$$

2. Bornes pour i_N. La valeur de l'indice i_N pour lequel le maximum de E_i ($2 \leqslant i \leqslant N-1$) est atteint, est telle que : $\frac{N+1}{4} \leq i_N \leq \frac{N+1}{2}$ (voir le lemme 6.23).
3. Borne maximale de E_{i_N}. Le lemme 6.24 nous donne une constante $C \leqslant 2$ telle que $\frac{N+1}{4} \leq i \leq \frac{N+1}{2}$, on a : $E_i \leq CN$.

La combinaison des deux derniers résultats achève la preuve. □

Restent à prouver les lemmes 6.23 et 6.24. En voici le détail :

La proposition 6.20 repose sur les lemmes 6.23 et 6.24, qui eux-mêmes reposent sur les lemmes 6.21 et 6.22.

Lemme 6.21. *Si $2 \leqslant i < \frac{N+1}{2}$, on a : $E_i - E_{N-i+1} > 0$.*

Démonstration. Soit $2 \leqslant i < \frac{N+1}{2}$. Si $j < i$ ou $j \geqslant N-i+1$, on a $b_{i,j} - b_{N-i+1,j} = 0$. Si $i \leqslant j < N-i+1$, on a $b_{i,j} - b_{N-i+1,j} = \frac{(i-1)(N-i)}{(j-1)(N-j)} > 0$. □

Lemme 6.22. *Si $3 \leqslant i \leqslant \frac{N+1}{2}$, on a :*

$$E_i - E_{i-1} = -1 + \frac{2(N-i+1)}{N-1} \sum_{i-1}^{N-i} \frac{1}{j}.$$

s

Lemme 6.23. *Soit i_N dans $\{2, ..., N-1\}$ telle que $E_{i_N} = \max\{E_i \, ; \, i \in \{2, ..., N-1\}\}$. Alors :*

$$\frac{N+1}{4} \leqslant i_N \leqslant \frac{N+1}{2}.$$

Démonstration. Grâce au lemme 6.21, on a $i_N \leqslant \frac{N+1}{2}$. Pour l'autre inégalité, on peut supposer $N \geqslant 8$.

– Soit i tel que $3 \leqslant i \leqslant \frac{N+1}{2}$; alors, $E_i > E_{i-1}$ si et seulement si $\sum_{i-1}^{N-i} \frac{1}{j} > \frac{N-1}{2(N-i+1)}$ (voir le lemme 6.22). Comme $i \mapsto \sum_{i-1}^{N-i} \frac{1}{j}$ est une suite décroissante de $\sum_1^{N-2} \frac{1}{j}$ vers $\frac{4(N-1)}{N(N-2)}$ ou $\frac{2}{N+1}$, selon la parité de N, et $i \mapsto \frac{N-1}{2(N-i+1)}$ est un suite croissante de $\frac{1}{2}$ à $\frac{N-1}{N+2}$ ou $\frac{N-1}{N+1}$ respectivement, il existe un indice l tel que $E_2 ... E_l$ est croissante et la suite $E_{l+1}, ...$ est décroissante. De plus, on doit avoir $l = i_N$ et i_N est caractérisé par :

$$\begin{cases} \forall i \leqslant i_N,\ E_i > E_{i-1}, \\ \forall i > i_N,\ E_i < E_{i-1}. \end{cases}.$$

– Soit i le plus grand entier plus grand ou égal à $\frac{N+1}{4}$ ($\frac{N+1}{4} \leqslant i \leqslant \frac{N+4}{4}$). Afin de prouver $\frac{N+1}{4} \leqslant i_N$, il suffit de démontrer que $E_i > E_{i-1}$.
$E_i - E_{i-1} = -1 + \frac{2(N-i+1)}{N-1} \sum_{i-1}^{N-i} \frac{1}{j} \geqslant -1 + \frac{3N}{2(N-1)} \sum_{i-1}^{N-i} \frac{1}{j} \geqslant -1 + \frac{3}{2} \sum_{i-1}^{N-i} \frac{1}{j}$.
Rappelons l'inégalité $\ln \frac{N-i+1}{i-1} \leqslant \sum_{i-1}^{N-i} \frac{1}{j}$ (utilisant $\frac{1}{j} \geqslant \ln \frac{j+1}{j}$). Ainsi : $E_i - E_{i-1} \geqslant -1 + \frac{3}{2} \ln \frac{N-i+1}{i-1} \geqslant -1 + \frac{3}{2} \ln \frac{3N}{N} = -1 + \frac{3}{2} \ln 3 > 0$.

□

Lemme 6.24. *Il existe $C > 0$ avec $0 < C \leq 2$ tel que, pour tout $N \in \mathbb{N}$ et tout i avec $\frac{N+1}{4} \leqslant i \leqslant \frac{N+2}{2}$, on a :*

$$E_i \leqslant CN.$$

Démonstration. Soit $N \geqslant 8$ et i tels que $\frac{N+1}{4} \leqslant i \leqslant \frac{N+1}{2}$. Alors :
$E_i = N \sum_{j=N-i+1}^{N-2} \frac{1}{j} + \frac{i(2N-i)}{N-1} \sum_{j=i-1}^{j=N-i} \frac{1}{j}$.
Rappelons que : $\sum_l^m \frac{1}{j} \leqslant \ln \frac{m}{l-1}$ (utilisant $\frac{1}{j} \leqslant \ln \frac{j}{j-1}$). Ainsi :

$$\begin{aligned} E_i &\leqslant N \ln \tfrac{N-2}{N-i} + \tfrac{(i-1)(2N-i)}{N-1} \ln \tfrac{N-i}{i-2} \\ &\leqslant N \ln \tfrac{2(N-2)}{N-1} + (\tfrac{N-1}{2} \tfrac{7N}{4} \tfrac{1}{N-1}) \ln \tfrac{3N-1}{N-7} \\ &\leqslant N(\ln 2 + \tfrac{7}{8} \ln 3). \end{aligned}$$

□

Chapitre 7

Coupling et auto-stabilisation

La difficulté avec la tolérance, vient de ce qu'elle apparaît à la fois nécessaire et impossible.
Bernard Williams

Ce chapitre est consacré à l'application des techniques de coupling pour des algorithmes informatiques sur des systèmes distribués. Nous nous intéresserons notamment aux algorithmes auto-stabilisants. Les définitions précises de toutes ces notions données dans les sections 7.1 et 7.2 rempliront la première partie du chapitre, et les déclinaisons du coupling et leurs applications à nos problèmes se trouveront dans les sections 7.4 et 7.5. Les notations que nous utiliserons ici sont principalement celle de la thèse de Marie Duflot [Duf03] et de l'article [FMP04b].

7.1 Systèmes distribués

Un *système distribué*, également appelé *système réparti*, se définit de la manière suivante. Il se compose d'un ensemble d'entités autonomes (ordinateurs, processeurs, automates, processus...) reliés au moyen de canaux de communication ou de variables partagées. Ils peuvent donc communiquer entre eux et évoluer en fonction de leur propre état et des informations échangées. On utilise souvent le formalisme des graphes pour décrire la topologie de ces systèmes. Ainsi, les entités sont appelées les *nœuds* du système et on utilise des arêtes pour représenter les communications entre les nœuds. Un exemple de système distribué est donné à la figure 7.1.

De nos jours, on ne met plus en doute l'utilité des systèmes distribués et des algorithmes qui leur sont associés. Ceux-ci sont présents dans des applications aussi diverses que les réseaux (qu'ils soient locaux, ou longue distance comme Internet), les ordinateurs multiprocesseurs, ou certains systèmes critiques dans lesquels on utilise plusieurs modules effectuant la même tâche pour augmenter la fiabilité. La section 7.1.2 présente les différents objectifs visés par les systèmes distribués, ainsi que les domaines associés à ces objectifs.

Toutes ces considérations sur les systèmes distribués, ainsi que de nombreux exemples de systèmes ou algorithmes distribués peuvent être trouvés dans [Tel91, Tel94, Lyn96].

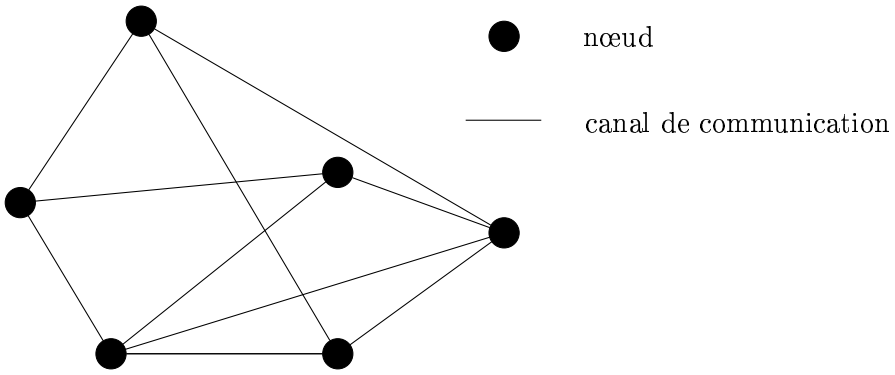


FIG. 7.1 – Un exemple de réseau associé à un système distribué.

7.1.1 Notions de base

Du fait que les systèmes distribués possèdent deux niveaux d'abstraction bien distincts (celui des processus et celui du système tout entier), on a besoin de notions relatives à ces deux niveaux.

On utilise la notion d'*état local*, ou tout simplement d'*état* pour désigner celui d'un processus. L'*état global* du système, c'est-à-dire la liste des états locaux des processus, est appelé *configuration*.

De même, pour caractériser l'évolution du système, on utilise le terme d'*action* pour désigner un pas d'évolution d'un processus, et le terme de *transition* pour un pas d'évolution du système global. Ainsi une transition correspond à ce qu'un (ou plusieurs) processus effectue(nt simultanément) une action.

7.1.2 Objectifs des systèmes distribués

Les situations impliquant les systèmes distribués sont très variées, et visent des buts différents. En voici les principaux :

- **L'échange d'informations :** Le besoin d'échanger des informations entre ordinateurs s'est développé dans les années soixante, alors que les grandes universités et les entreprises se dotaient de leurs propres serveurs. La coopération entre ces différentes organisations a été facilitée par l'échange de données entre ordinateurs, et elle a encouragé le développement de réseaux longue distance. Aujourd'hui ces réseaux sont partout, que ce soit à grande échelle (Internet reliant les différentes régions du monde) ou à plus petite échelle (Intranet reliant les différents ordinateurs d'une même entreprise). Ils permettent aux particuliers comme aux professionnels d'échanger des données, par exemple en téléchargeant des fichiers ou par courrier électronique.
- **Le partage des ressources :** Si les ordinateurs se sont répandus au point que de nombreuses personnes en sont équipées dans une entreprise, ce n'est pas le cas de tous les périphériques (imprimantes, unités de stockage,...) Il est alors judicieux de connecter des machines au sein d'un réseau local, pour leur permettre d'accéder à ces périphériques partagés.

- **Augmentation de la fiabilité :** Dans un système distribué, du fait de l'existence de petites entités (les nœuds du système), il est souhaitable que, alors que certains nœuds sont défaillants, d'autre continuent à fonctionner en prenant en charge le travail des nœuds défaillants. Ainsi, si l'on fait effectuer la même tâche par plusieurs processeurs distincts, en prenant en sortie le résultat le plus probable (i.e. obtenu par le plus grand nombre de processeurs), on augmente la fiabilité grâce à la répartition.
- **Augmentation des performances :** Le fait de disposer de plusieurs processeurs, que ce soit au sein d'une même machine ou *via* un réseau, peut permettre de partager une tâche à effectuer entre plusieurs processeurs, et de les faire tourner en parallèle. Ainsi le temps global nécessaire pour effectuer la tâche peut être diminué d'autant.

Dans le cadre de l'augmentation de la fiabilité et des performances, on considère surtout des machines multiprocesseurs. Il arrive également que des ordinateurs distants se mettent à participer à une même tâche, comme par exemple le projet RC5[1] qui consistait à trouver une clef de 64 bits en tentant toutes les solutions possibles. Ce projet a été mené à bien en près de 5 ans, et au total par plus de 300000 personnes. Les échanges d'information se font sur les réseaux, que ce soient les réseaux longue distance ou les réseaux locaux. Le partage des ressources se fait quant à lui principalement à l'échelle des réseaux locaux, mais pas uniquement.

7.1.3 Problèmes soulevés par les systèmes distribués

Ces systèmes distribués apportent d'une part des solutions nouvelles ou parfois plus rapides à de nombreux problèmes et soulèvent d'autre part de nouvelles questions. Les systèmes distribués ont besoin de méthodes particulières pour être étudiés car ils sont relativement différents des systèmes classiques.

Tout d'abord il est courant en algorithmique distribuée que chaque entité ne connaisse qu'une partie de l'état du système, contrairement au cas classique où l'on a une connaissance globale de celui-ci. Cette hypothèse est tout à fait raisonnable. En effet, on imagine mal, dans un système comportant de nombreux processus, que chacun d'entre eux maintienne une connaissance globale du système. Cela demanderait une place mémoire trop importante, ou beaucoup d'échanges de messages. Dans le cas distribué on doit donc réussir à accomplir des tâches en n'ayant qu'une connaissance partielle du système.
Lorsqu'on veut, dans le cas de l'exclusion mutuelle, s'assurer qu'un seul processus du réseau possède un certain privilège, la difficulté vient justement du fait qu'il est impossible, pour n'importe quel processus, de savoir si, à l'autre bout du réseau, un autre processus possède ou non un privilège.

Un autre point important est l'absence de temps global sur le système. Chaque processus autonome peut *a priori* évoluer selon une échelle de temps qui lui est propre. Ainsi on n'a pas vraiment de contrôle permettant de savoir dans quel ordre les processus vont agir. Certains événements peuvent même avoir lieu de manière simultanée. Ainsi, pour formaliser cette notion d'ordre ou de simultanéité des actions, on va avoir besoin d'un modèle, celui de politique défini au chapitre 2 (ou aussi de *démon* ou scheduler en anglais).

[1] pour plus d'informations, regarder la page web : `http ://www.distributed.net/rc5/`

7.1.4 Systèmes distribués probabilistes

Les probabilités ont été introduites, dans les systèmes distribués comme ailleurs, pour principalement deux raisons. D'un côté elles permettent de résoudre plus efficacement certains problèmes, et d'un autre côté elles permettent de résoudre d'autres problèmes dont on ne connaissait pas de solution dans le cadre déterministe, voire dont on avait prouvé qu'il n'existait pas de solution déterministe. Un exemple de ce dernier cas est l'algorithme du dîner des philosophes, pour lequel il a été prouvé qu'il n'existait pas de solution déterministe, symétrique et totalement distribuée (voir [RL94]).

Aussi, il a fallu adapter le cadre des systèmes distribués à l'utilisation de choix probabilistes.
Lorsque l'on introduit des probabilités dans un système distribué, on retrouve le modèle de processus de décision markovien décrit à la section 2.3. Dans ce cas, c'est la politique qui est source de non-déterminisme.
Les actions du processus de décision markovien correspondent ici aux choix non déterministes de la politique entre les (sous-ensembles de) positions activables. Les distributions de probabilités que l'on avait dans le processus de décision markovien correspondent ici aux choix probabilistes entre les différents résultats possibles des règles probabilistes.

Exemple 7.1. *Considérons un système distribué probabiliste à N processus disposés en anneau, dont l'espace d'états est $\Sigma = \{0, 1, 2\}$, et défini par les deux règles probabilistes suivantes :*

$$01 \rightarrow \begin{cases} 11 \text{ } \textit{avec probabilité } 1/2 \\ 21 \text{ } \textit{avec probabilité } 1/2 \end{cases}$$
$$02 \rightarrow \begin{cases} 12 \text{ } \textit{avec probabilité } 1/3 \\ 22 \text{ } \textit{avec probabilité } 2/3 \end{cases}$$

Supposons que le système possède un démon distribué. Partant de la configuration $x =$ 01020, il y a deux processus activables, en positions 0 et 2. Si le démon sélectionne ces deux positions, alors la distribution de probabilité sur l'ensemble des configurations suivantes possibles est :

11120	*avec probabilité 1/6*
11220	*avec probabilité 1/3*
21120	*avec probabilité 1/6*
21220	*avec probabilité 1/3*

♣

La définition de l'espace de probabilité se fait grâce aux définitions données dans le chapitre 2 que ce soit pour les systèmes purement probabilistes (ceux où une seule politique est possible) qui sont modélisés par des chaînes de Markov comme pour les systèmes comprenant du non déterminisme modélisés par des processus de décision markoviens.

7.1.5 Systèmes distribués en anneau

Dans la suite, sauf quand cela sera précisé autrement, on va s'intéresser au cas particulier où les différents processeurs sont disposés en anneau. On dispose donc d'un nombre paramétré N de machines, par exemple N automates finis, et pour effectuer une transition, chaque automate peut par exemple prendre en compte, outre son état propre, celui de son (ses)

voisin(s) immédiat(s). L'état de chaque processeur est donc considéré comme une variable partagée, sur laquelle seul le processeur lui-même possède le droit d'accès en écriture, mais son(ses) voisin(s) possèdent un droit d'accès en lecture.

Cet exemple est illustré dans la figure 7.2, où une flèche d'un processeur P vers un processeur Q signifie que Q peut lire l'état de son voisin P. Dans ce premier exemple, on aurait pu ne pas représenter les flèches sur les transitions, du fait que le réseau est bidirectionnel.

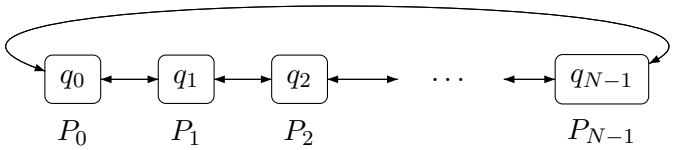


FIG. 7.2 – Système en anneau où chaque processeur lit l'état de ses deux voisins.

On peut aussi imaginer le cas d'un réseau unidirectionnel où chaque processeur ne peut lire que l'état de son voisin de gauche (figure 7.3). Selon la définition de voisin, le processeur à sa droite n'est donc pas son voisin.

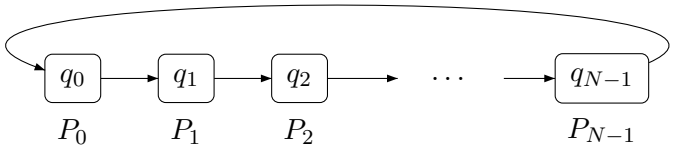


FIG. 7.3 – Système en anneau où chaque processeur peut lire l'état de son voisin de gauche.

Notons que ce sont des topologies de ce type que nous avons étudiées dans le cadre de l'ASEP dans le chapitre 6.
Il nous arrivera de considérer d'autres topologies (linéaire, graphes) que nous définirons plus tard.

7.1.6 Algorithmes distribués probabilistes vus comme des chaînes de Markov

Rappelons que l'ensemble des configurations du système est noté Ω, et que nous entendons par politique (défini au chapitre 2) l'ordonnancement qui consiste à choisir à chaque étape quelles machines seront mises à jour (parfois appelé démon ou scheduler).
Dans ce chapitre, nous considérerons que la politique est fixée et sans mémoire : à chaque étape la politique consiste en le choix d'un sous ensemble de machines ne dépendant que de la configuration actuelle.
Par exemple, nous considérerons le cas d'une politique *synchrone* (resp. *centrale aléatoire*) qui sélectionne à chaque étape toutes les machines (resp. une seule choisie de manière aléatoire et uniforme).
Une fois qu'une machine est sélectionnée, son état (et parfois celui de ses voisins) se met à jour en suivant l'algorithme.
Pour une politique sans mémoire et fixée un algorithme distribué probabiliste peut être modélisé comme une chaîne de Markov $\mathcal{A}$ sur Ω (voir [DFP01b]) : la probabilité, à chaque étape,

pour se rendre d'une configuration x à une autre y est une constante, noté $\mathbf{P}(x, y)$, qui ne dépend que de x et y.
Supposons que toutes les configurations de Ω sont de la forme x_e $e \in \{1, 2, \cdots, |\Omega|\}$. Ainsi, $\mathcal{A}$ est caractérisée par la matrice de transition de la chaîne de Markov associée $\mathbf{P}$ sur $\Omega \times \Omega$ qui a $\mathbf{P}(x_e, x_f)$ pour (e, f)-coordonnée. La probabilité pour se rendre de x à y en t étape est $\mathbf{P}^t(x, y)$.

Par la suite, une fois que la politique est fixée nous nous servirons de cette modélisation pour confondre un algorithme distribué probabiliste et sa chaîne de Markov associée.

7.2 Auto-stabilisation

7.2.1 Tolérance aux fautes

Comme tout système, les systèmes distribués sont susceptibles de subir des pannes (ou fautes), à savoir que tout ou partie du système peut, à un moment donné, cesser de réagir suivant sa spécification. Cette notion de faute recouvre des cas assez divers comme la corruption de code, les problèmes de transmission de messages, le ralentissement d'un processeur, voire son arrêt intempestif.

Un système tolérant aux fautes devrait, idéalement, pouvoir continuer à fonctionner correctement dans de telles situations. Si les fautes peuvent arriver n'importe quand et être de n'importe quelle nature, le problème devient vite très complexe.

Selon le type de faute possible, les façons d'y remédier sont diverses.
Si par exemple les fautes se situent au niveau de la transmission des messages, qui peuvent être légèrement déformés au cours de la transmission, il est possible d'utiliser des codes détecteurs d'erreur. Le principe est d'envoyer, en plus du message, une part d'information supplémentaire, qui est fonction du message. Le receveur peut alors récupérer les deux parties, recalculer la fonction, et si le résultat qu'il obtient n'est pas conforme à l'information supplémentaire envoyée, il détecte l'erreur et peut demander qu'on lui envoie le message de nouveau.
Si l'on veut pouvoir retrouver le message à partir du message modifié, on peut utiliser des codes correcteurs d'erreur. L'information est alors envoyée de manière redondante, si bien que, lorsque la quantité d'erreurs n'est pas trop grande, on peut reconstituer le message original directement à partir du message déformé.

Les messages peuvent aussi, suivant le modèle que l'on considère, être perdus, voire reçus plusieurs fois au lieu d'une, et il est parfois nécessaire de détecter ces problèmes. Par exemple si un ordre de débiter un compte en banque est reçu deux fois au lieu d'une, il ne faut pas que l'opération soit effectuée deux fois.

Les principaux modèles de pannes considérés sont les suivants :

- on dit d'une panne qu'elle est *transitoire* lorsqu'elle n'intervient qu'une seule fois, et que, lorsqu'elle se termine, les processeurs reprennent une activité normale ;
- une panne est dite *définitive* (ou *fatale*) lorsqu'un (ou plusieurs) processeur est(sont) définitivement arrêté(s) (on utilise le terme de *crash* en anglais) ;
- enfin les pannes *byzantines* sont celles qui sont totalement incontrôlables. Un processeur peut s'arrêter de fonctionner, puis reprendre son activité.

Le qualificatif de "byzantin" vient d'un article de Lamport, Shostak et Pease [LSP82], présentant un problème de consensus sous la forme de discussions, par le biais de messagers, entre des généraux de l'armée byzantine, devant décider d'une éventuelle attaque commune contre une ville ennemie. La difficulté de ce problème vient du fait que certains des généraux

peuvent être des traîtres, et donc agir à leur guise, envoyant des mauvaises informations, voire des informations contradictoires.

Selon le type de panne, il faut concevoir des algorithmes soit pour les gérer, et donc continuer à fonctionner correctement même pendant la panne (sous réserve que celle-ci ne soit pas trop étendue), soit pour s'en remettre, à savoir permettre au système, après la panne, de revenir à une situation et un comportement corrects.

En ce qui concerne cette deuxième possibilité, et lorsque le retour à un comportement normal se fait sans aide extérieure, on parle d'auto-stabilisation. Nous donnerons par la suite plusieurs exemples d'algorithmes auto-stabilisants, en nous intéressant notamment à la preuve de la correction de ces algorithmes et aussi à l'étude du temps qu'il leur faut pour permettre au système de retrouver un comportement normal (on parle de temps de convergence).

7.2.2 Définitions

L'auto-stabilisation est un concept introduit par Dijkstra en 1973 (voir les articles [Dij73, Dij74]), qui caractérise des systèmes capables par eux-mêmes de revenir, après une panne ponctuelle, dans un état correct. Pour une présentation en détail du sujet, de ses enjeux et des problèmes que l'auto-stabilisation permet de résoudre, on peut se référer au livre de Dolev [Dol00].

L'auto-stabilisation est un cas particulier de la convergence. En effet, elle assure que, quel que soit l'état de départ, après une certaine phase dite de stabilisation, le système va arriver dans ensemble de "bons" états, appelé ensemble *légitime* (voir définition 7.2) et à partir de ce moment là, le système va avoir un comportement "correct" (conforme à une certaine spécification).

Définition 7.2. *[Tel94] Soit P une spécification sur les exécutions. On dit qu'un système de transitions $\mathcal{S}$ est* auto-stabilisant *vers la spécification P s'il existe un ensemble $\mathcal{L}$ de configurations dites* légitimes *tel que :*

- *d'une part le système $\mathcal{S}$ converge vers $\mathcal{L}$ (***convergence***) et,*
- *d'autre part, toute exécution partant d'une configuration de $\mathcal{L}$ satisfait P (***correction***).*

Comme on le voit dans la définition, un système n'est pas auto-stabilisant en soi, mais pour une spécification particulière. Cependant, dans la suite, quand il n'y aura pas d'ambiguïté sur l'ensemble légitime, on pourra l'omettre.

Dans toute la suite de cette thèse, nous allons étudier des algorithmes pour lesquels la spécification P est du genre "on reste toujours dans l'ensemble de configurations E". De plus, pour tous les algorithmes considérés l'ensemble E choisi va être clos, à savoir :

Définition 7.3. *Étant donné un ensemble E et une relation de transition $\rightarrow$, on dit que E est* clos *pour $\rightarrow$ si, pour tout élément e de E on a : $e \rightarrow e'$ implique $e' \in E$*

Dans ce cas, pour montrer l'auto-stabilisation du système $\mathcal{S}$ vers P, on va montrer la convergence de $\mathcal{S}$ vers E, et bien sûr vérifier la clôture de E pour le système $\mathcal{S}$. Ainsi on va montrer que l'ensemble de configurations $\mathcal{L} = E$ est légitime. Par abus de langage, nous allons également dire que le système $\mathcal{S}$ est auto-stabilisant vers $\mathcal{L} = E$, et non plus vers la spécification P qui qualifie l'ensemble des exécutions qui restent dans E.

Avant de conclure cette section, il est important d'ajouter une propriété (d'ailleurs évoquée par Dijkstra à l'origine [Dij74]), et qui nous permettra de distinguer une certaine classe d'algorithmes auto-stabilisants.

Définition 7.4. *On dira qu'un algorithme $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$ au sens de Dijsktra ssi :*

1. *$\mathcal{L}$ est fortement connexe,*
2. *$\mathcal{L}$ est clos, et,*
3. *$\mathcal{A}$ converge vers $\mathcal{L}$.*

Notons que les deux dernières hypothèses correspondent à celle de la définition 7.2

Remarque :

La définition originelle de Dijkstra n'est pas donnée en ces termes. Tout d'abord sa présentation se fait sur un graphe de machines où les machines placées sur des noeuds connectés sont appelées voisins. De plus, il définit la notion de *privilège* comme une fonction booléenne de l'état d'une machine et de ses voisins indiquant si une transition est possible pour cette machine. Sous ces notations il définit l'auto-stabilisation de la manière suivante :

(a) Dans tous les états de $\mathcal{L}$ il y a toujours au moins un privilège,
(b) $\mathcal{L}$ est clos,
(c) Chaque privilège doit être présent dans au moins un des états de $\mathcal{L}$,
(d) Pour tout couple d'états de $\mathcal{L}$, il existe une suite de transitions menant de l'un à l'autre,
(e) $\mathcal{A}$ converge vers $\mathcal{L}$.

La condition (b) correspond à notre condition 2 et la condition (e) à la 3. De plus, le fait que nos systèmes vont se comporter commes des chaînes de Markov assurera que l'algorithme ne terminera jamais, ainsi la condition (a) (équivalente au fait que l'algorithme ne termine pas) est vérifiée. Les conditions 1 et (d) sont équivalentes, en effet la condition (d) donne la définition de la forte connexité. Enfin, la condition (c) est souvent considérée comme inutile (voir par exemple [Sch93]), c'est pour cette raison que nous l'avons omise. Notre définition correspond donc à l'auto-stabilisation au sens de Dijkstra.

7.2.3 Convergence probabiliste

Dans toute la suite, nous n'étudierons que des algorithmes probabilistes, il convient donc de donner la définition formelle de la convergence probabiliste.

Étant donnés un système $\mathcal{S}$, une politique $\mathcal{O}$ et un ensemble de configurations $\mathcal{L} \subseteq X$, la propriété de convergence probabiliste de $\mathcal{S}$ vers $\mathcal{L}$, habituellement notée

$$\forall x \in X,\ \mathbb{P}_{x,\Omega}(\sigma = x_0, J_0, ..., x_n, J_n, ... \in \Omega_{x,\mathcal{O}} | \exists i :\ x_i \in \mathcal{L}) = 1$$

(où chaque J_i est un ensemble d'actions activables dans la configuration x_i choisi par $\mathcal{O}$) sera abrégée en :

$$\forall x \in X,\ \ \mathbb{P}(x \xrightarrow[\mathcal{S}]{\mathcal{O}}{}^{*} \mathcal{L}) = 1.$$

Donnons à présent le résultat liant les chaînes de Markov aux algorithmes auto-stabilisants au sens de Dijsktra.

Proposition 7.5. *Un algorithme distribué probabiliste (i.e. modélisable comme une chaîne de Markov) $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$ au sens de Dijsktra si et seulement si $\mathcal{L}$ est l'unique ensemble ergodique (ou composante fortement connexe close) de $\mathcal{A}$ vu comme une chaîne de Markov.*

Démonstration. Supposons que $\mathcal{L}$ est l'unique ensemble ergodique de $\mathcal{A}$. Ainsi $\mathcal{L}$ satisfait les propriétés de clôture et forte connexité par définition, de plus d'après le théorème de Markov 2.21, la chaîne de Markov converge vers la réunion des ensembles ergodiques et donc vers $\mathcal{L}$. Ainsi $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$
Supposons à présent que $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$, de ce fait $\mathcal{L}$ est clos et fortement connexe par définition, il est donc un ensemble ergodique. Montrons le résultat par l'absurde, supposons qu'il existe un second ensemble ergodique disjoint $\mathcal{L}'$. Si on prend $y \in \mathcal{L}'$, toutes les transitions partant de y restent en $\mathcal{L}'$ car $\mathcal{L}'$ est clos. La probabilité d'atteindre $\mathcal{L}$ est donc nulle ce qui contredit la convergence vers $\mathcal{L}$. $\mathcal{L}$ est donc le seul ensemble ergodique. □

Intérêt de l'auto-stabilisation

L'auto-stabilisation est une propriété qui est recherchée dans le cadre de la tolérance aux pannes. Par définition, un système auto-stabilisant a la propriété que, quel que soit l'état de départ, il revient en un temps fini dans un ensemble de configurations dites légitimes, et il n'en sort plus. Ainsi, si l'on suppose qu'à un moment donné, suite à une panne transitoire, le système se trouve dans une configuration illégitime, on sait que, en l'absence de nouvelle panne, il va retourner dans une configuration légitime en un temps fini, puis rester dans l'ensemble légitime.

On parle d'*auto*-stabilisation car ce retour à un comportement normal se fait sans aucune aide extérieure, que ce soit pour détecter la panne ou pour y remédier.

Il faut tout de même préciser que cette auto-stabilisation ne peut avoir lieu que tant que le code du programme et le comportement des différents processeurs ne seront pas affectés après la panne. On suppose donc que cette panne n'a fait, en fin de compte, que changer les états des processeurs qui, une fois la panne finie, se remettront à exécuter normalement le code.

Pour plus d'informations sur l'auto-stabilisation en général, on peut se rapporter en français à la thèse de Sylvie Delaët [Del95]. En anglais, on peut citer l'article de synthèse de Schneider [Sch93] ou le livre de Dolev [Dol00].

Avant de conclure cette section, donnons un exemple simple d'algorithme probabiliste auto-stabilisant (un algorithme d'exclusion mutuelle) : l'algorithme de Herman. Nous allons par la suite étudier cet algorithme en détail, il peut être vu comme une chaîne de Markov car il n'y a qu'une seule politique possible, toutes les actions pour toutes les machines se passent en même temps, on parle d'algorithme synchrone.

Exemple 7.6. (L'algorithme de Herman)
L'ensemble d'états est $\Sigma = \{0, 1\}$, et le nombre de machine N est impair disposé sur un anneau unidirectionnel et la politique est synchrone (toutes les machines sont mises à jour à chaque étape).
Nous noterons $X_t = (x_t(1), x_t(i), ..., x_t(N))$ une configuration du réseau après t étapes de l'algorithme, dans lequel $x_t(i)$ représente l'état courant de la machine i. L'algorithme est donné par les transitions suivantes :

- *Si $x_t(i) \neq x_t(i-1)$ alors $x_{t+1}(i) = 1 - x_t(i)$*

– *Si* $x_t(i) = x_t(i-1)$ *alors* $x_{t+1}(i) = \begin{cases} 1 \text{ avec probabilité } 1/2 \\ 0 \text{ avec probabilité } 1/2 \end{cases}$

On peut tout d'abord remarquer que, même si les règles déterministes consistent à copier l'état du voisin de gauche, elles n'ont pas pour effet d' "uniformiser" les états des machines. Comme, pour chaque transition, on applique une règle à chaque position, si on a dans une configuration x *une suite de la forme* 01010101*, elle va se transformer en* $a0101010$ *où* a *dépend de la lettre qui précédait le premier* 0 *de ce motif dans la configuration. L'application des règles non probabilistes n'uniformise donc pas l'état des différents processeurs, mais préserve plutôt ces suites alternées.*

Pour chaque configuration, toutes les positions sont activables : lorsqu'un processeur est dans le même état que son voisin de gauche, on peut lui appliquer une règle probabiliste, et s'ils sont dans des états différents, on peut lui appliquer une règle déterministe.

Si l'on considère par exemple la configuration $x = 00101$*, comme la politique va sélectionner tous les processeur, on va appliquer une règle probabiliste (qui va modifier (ou non) la lettre à la position 1) et quatre règles déterministes (partout ailleurs). On a donc deux possibilités de transition :* $x \to x_1 = 11010$ *et* $x \to x_2 = 10010$*, chacune de probabilité 1/2.*

Enfin, nous appellerons jeton une suite de deux machines ayant le même état, par exemple dans la configuration 00101 *il n'y a qu'un seul jeton entre les machines* 1 *et* 2. ♣

Cet exemple nous permettra d'illustrer les théorèmes donnés dans les prochaines sections de manière simple.

7.2.4 Preuves classiques de convergence

Les preuves classiques pour prouver la convergence des algorithmes auto-stabilisants probabilistes consistent en la détermination d'une distance sur l'ensemble des configurations qui peut décroître en une étape. Nous reprendrons ici les théorèmes de la thèse de M. Duflot [Duf03] (les preuves se trouvent dans la thèse et dans [DFP01b]).
Nous noterons par $(X_t)_{t\in\mathbb{N}}$, soit la chaîne de Markov, soit le processus de décision markovien correspondant à l'algorithme étudié.
Avant de donner les résultats principaux, on peut rappeler qu'ils découlent du théorème 2.21 quand l'algorithme peut être vu comme une chaîne de Markov ou du théorème 2.39 lorsqu'il est vu comme un processus de décision markovien.

Théorème 7.7. *Étant donné un algorithme* $\mathcal{A}$ *modélisé par un processus de décision markovien* (S, A, p) *, supposons qu'il existe une fonction* D *et un ordre* $\ll$ *tels que*
Prop : $\forall X \notin \mathcal{L}\ \forall i \in A(x)\ \exists Y :\ (X \xrightarrow[\mathcal{A}]{i} Y \wedge (Y \in \mathcal{L} \vee D(Y) \ll D(X))))$
alors pour toute politique $\mathcal{O}$*,* $\forall X \quad \mathbb{P}(X \xrightarrow[\mathcal{A}]{\mathcal{O}}{}^{*} \mathcal{L}) = 1.$

Ce théorème peut évidemment se décliner dans le cas où l'algorithme étudié n'autorise qu'une seule politique (une chaîne de Markov).

Théorème 7.8. *Étant donné un algorithme* $\mathcal{A}$ *modélisé par une chaîne de Markov, s'il existe une fonction* D *et un ordre* $\ll$ *tels que*
Prop' : $\forall X \notin \mathcal{L}\ \exists Y :\ (X \xrightarrow[\mathcal{A}]{} Y \wedge (Y \in \mathcal{L} \vee D(Y) \ll D(X)))$

alors $\forall X \quad \mathbb{P}(X \xrightarrow[\mathcal{A}]{*} \mathcal{L}) = 1.$

Dans [Duf03, DFP01b] on peut trouver de nombreux exemples d'applications de ces théorèmes dans le cadre des processus de décision markoviens ou des chaînes de Markov.

7.3 Temps de convergence

Après avoir montré la convergence des algorithmes auto-stabilisants il est crucial de pouvoir déterminer le temps de convergence de ces algorithmes, à savoir le nombre d'étapes nécessaires (nombre de transitions) avant l'arrivée dans l'ensemble des états légitimes.
Pour cela nous allons définir deux notions de temps de convergence, l'une (le temps de hitting) est celle couramment utilisée par les personnes du domaine de l'auto-stabilisation, l'autre le temps de ε-absorption découle du temps de mélange et peut être parfois utile.
Dans cette section nous donnerons ainsi les différentes méthodes classiques de calcul du temps de convergence.

Définition 7.9. *Soit une chaîne de Markov $\mathcal{A}$ et un ensemble $\mathcal{L}$, le temps de hitting de $\mathcal{L}$ est le temps d'arrêt défini par :*

$$\mathbf{H}_{\mathcal{L}} = max_{x \in \Omega}\ E(H_{x\mathcal{L}}),$$

où nous noterons $H_{x\mathcal{L}} = min\{t :\ X_t \in \mathcal{L} \mid X_0 = x\}$.

Nous pouvons donner ici deux propositions nécessaires pour la suite :

Proposition 7.10. *Etant donné un ensemble clos $\mathcal{L}$, si $\mathbf{H}_{\mathcal{L}}$ est fini, alors $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$.*

Démonstration. Supposons que $\mathcal{A}$ ne converge pas vers $\mathcal{L}$ il existerait une exécution X_t et un réel positif a tels que $\forall t \in \mathbb{N} \quad \mathbb{P}[X_t \in \mathcal{L}] \geq a$ de ce fait $H_{x\mathcal{L}} = \infty$ ce qui est une contradiction. □

Proposition 7.11. *Etant donné un ensemble ergodique $\mathcal{L}$, si $\mathbf{H}_{\mathcal{L}}$ est fini, alors $\mathcal{L}$ est l'unique ensemble ergodique, et $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$ (au sens de Dijsktra).*

Démonstration. La preuve de cette proposition est essentiellement la même que celle de la proposition précédente car un ensemble ergodique est clos par définition et qu'un algorithme auto-stabilisant ne peut converger que vers un seul ensemble ergodique. (Le temps de hitting serait infini si on partait de l'autre ensemble ergodique disjoint). □

Définition 7.12. *Soit une chaîne de Markov $\mathcal{A}$ et un ensemble ergodique $\mathcal{L}$, le temps d'ε-absorption par $\mathcal{L}$ est défini par :*

$$\mathbf{\Theta}_{\mathcal{L}}(\varepsilon) = max_{x \in \Omega}\ \Theta_{x\mathcal{L}}(\varepsilon),$$

où $\Theta_{x\mathcal{L}}(\varepsilon) = min\{t :\ P(X_t \in \mathcal{L}) \geq 1 - \varepsilon \mid X_0 = x\}$.

Remarquons que la notion de temps d'ε-absorption est voisine de celle de temps de mélange. En effet, dans le cas où $\mathcal{L}$ est réduit à un seul élément elles sont équivalentes ; dans les autres cas il s'agit d'une minoration du temps de mélange (en effet on n'attend pas que la distribution soit conforme à la distribution à la distribution stationnaire sur $\mathcal{L}$, on attend juste d' être en elle).

La suite de la section sera consacrée aux rappels des techniques classiques pour calculer le temps de convergence. Les notions essentielles à ce sujet peuvent se trouver dans [Bré99, AFar] (notamment les comparaisons possibles entre les différents temps de convergence dans le cas des chaînes de Markov irréductibles).

7.3.1 Calcul des valeurs propres

La première méthode qui vient à l'esprit pour calculer le temps de convergence d'une chaîne de Markov est celle de l'étude de la matrice stochastique associée et au calcul des valeurs propres. On peut trouver le détail des relations entre la deuxième valeur propre (rayon spectral) et les différents temps de convergence dans [AFar, Bré99] par exemple.
Le problème essentiel lié à ces techniques est que la matrice de la chaîne de Markov est souvent dans un espace de taille immense (dans nos exemples on a 2^N) et qu'il est souvent impossible de les calculer ni même de décrire simplement les matrices.
Dans [Duf03, DFP01b], on peut remarquer une technique astucieuse qui consiste à rassembler entre eux (lumping) des configurations qui se comporteront de la même manière par rapport à une certaine fonction sur l'ensemble des configurations φ. Cette méthode par lumping a permis de résoudre un grand nombre de problème de vitesse de convergence, mais nécessite de bien connaître les algorithmes étudiés car il faut à la fois trouver une distance qui convienne et rassembler les configurations de manière astucieuse.

7.3.2 Martingales

Une deuxième technique est utilisée dans [DGG$^+$02] pour la majoration du temps d'ε-absorption dans le cas de l'algorithme du dilemme du prisonnier itéré. Cette technique consiste en la découverte (encore) d'une fonction sur l'ensemble des configurations, puis à l'utilisation de théorèmes venus de la théorie des martingales (qui exploitent la décroissance en moyenne) pour majorer le temps qu'il faut à cette distance pour rejoindre 0.
Cette technique est souvent utilisée dans le cas de l'étude des marches aléatoires.

Nous présenterons donc une méthode qui parait à la fois plus simple et plus efficace et basée sur la technique de coupling expliquée au chapitre 6.

7.4 Preuves par coupling

Rappelons que les définitions du coupling sont données dans le chapitre 6
Le temps de coupling est souvent calculé comme une borne supérieure du temps de mélange, pour montrer la propriété de mélange rapide pour une chaîne de Markov $\mathcal{A}$ (le fait que le temps de mélange est polynomial en N et $\ln(\frac{1}{\varepsilon})$).
On va montrer dans cette section en utilisant les notations de [FMP04b] que le temps de coupling peut aussi donner une borne supérieure du temps de hitting.

Théorème 7.13. *Soit une chaîne de Markov $\mathcal{A}$ et un ensemble ergodique $\mathcal{L}$, s'il existe un coupling de temps de coupling fini* $\mathbf{T}$*, alors :*

1. *Le temps de hitting* $\mathbf{H}_{\mathcal{L}}$ *vérifie :* $\mathbf{H}_{\mathcal{L}} \leq \mathbf{T}$.
2. *$\mathcal{L}$ est l'unique ensemble ergodique, et $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$.*

Démonstration. Supposons qu'il existe un coupling de temps moyen de coupling fini $\mathbf{T}$, et montrons les conclusions 1 et 2.

1. Rappelons que : $H_{x\mathcal{L}} = min\{t : \ X_t \in \mathcal{L} \mid X^0 = x\}$, et $T_{xy} = min\{t : \ X_t = Y_t \mid X^0 = x, Y^0 = y\}$. Supposons à présent que $y \in \mathcal{L}$. Alors $Y_t \in \mathcal{L}$ car $\mathcal{L}$ est clos. Ainsi : $H_{x\mathcal{L}} \leq T_{xy}$ pour tout $x \in \Omega, y \in \mathcal{L}$. Et en prenant les espérances, puis les maxima des deux côtés, on obtient : $\mathbf{H}_{\mathcal{L}} \leq \mathbf{T}$.
2. L'unicité de $\mathcal{L}$ et l'auto-stabilisation de $\mathcal{A}$ se déduisent de la finitude de $\mathbf{H}_{\mathcal{L}}$ (conclusion 1) par la proposition 7.11.

□

7.4.1 Théorèmes

D'après le théorème 7.13, trouver une borne supérieure du temps de coupling T nous permet à la fois de prouver l'auto-stabilisation de l'algorithme et de majorer le temps de hitting de ce même algorithme.
Se basant sur des résultats classiques sur le temps de mélange (voir [DG98]), nous donnons dans cette section deux condition suffisantes pour majorer le temps de coupling. Dans chaque cas, cela nous permet non seulement de majorer le temps de hitting, mais cela nous apporte en plus une borne supérieure du temps d'ε-absorption.

Théorème 7.14. *Soit une chaîne de Markov $\mathcal{A}$ et un ensemble ergodique $\mathcal{L}$. Supposons qu'il existe un coupling (X_t, Y_t) et une fonction δ à valeurs entières sur $\Omega \times \Omega$ qui prend ses valeurs dans $\{0, 1, \cdots, B\}$ et telle que $\delta(X_t, Y_t) = 0$ ssi $X_t = Y_t$, et :*
$\exists \beta < 1 \ \ \forall (X_t, Y_t) \ \ E(\delta(X_{t+1}, Y_{t+1})) \leq \beta \delta(X_t, Y_t).$ $(*)$
Alors $\mathcal{L}$ est l'unique ensemble ergodique et $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$. De plus :

1. *Le temps de hitting vérifie :* $\mathbf{H}_{\mathcal{L}} \leq \frac{B}{1-\beta}$.
2. *le temps d'ε-absorption vérifie :* $\mathbf{\Theta}_{\mathcal{L}}(\varepsilon) \leq \frac{\ln(B/\varepsilon)}{1-\beta}$.

Démonstration. La preuve du théorème 7.14 repose sur la proposition suivante qui peut facilement se démontrer à l'aide de la théorie sur les surmartingales, en appliquant le théorème d'arrêt optionnel de Doob (voir par exemple [LRS95, DG98]) :

Proposition 7.15. *Supposons que $D = (D_0, D_1, \cdots)$ est un processus stochastique à valeurs positives sur $\{0, 1, \cdots, B\}$ tel que $E[D_{t+1}] \leq \beta D_t$ (avec $0 < \beta < 1$).*

Alors si τ est le premier temps tel que $D_t = 0$, on a : $E[\tau] \leq B/(1-\beta)$.

Preuve du théorème 7.14.
Considérons une fonction δ satisfaisant aux conditions du théorème 7.14, et montrons les résultats 1 et 2. (Le fait que $\mathcal{L}$ est l'unique ensemble ergodique, et que $\mathcal{A}$ est auto-stabilisant se déduisent de la conclusion 1, par la proposition 7.11.)

1. Considérons deux éléments $x, y \in \Omega$, et le coupling (X_t, Y_t) de configurations initiales $(X_0, Y_0) = (x, y)$. Soit D_t le processus stochastique défini par $D_t = \delta(X_t, Y_t)$ pour $t \geq 0$. Comme $\delta(X_t, Y_t) = 0$ ssi $X_t = Y_t$, la quantité $T_{x,y}$ est le temps nécessaire à D_t pour atteindre 0. Considérons le coupling (X_t, Y_t) partant de $(X_0, Y_0) = (x, y)$. Par la proposition 7.15, on a, pour tout $x, y \in \Omega$, $E(T_{x,y}) \leq B/(1-\beta)$.
   Aussi, par le théorème 7.13, on déduit :

$$\mathbf{H}_{\mathcal{L}} \leq max_{x,y}\ E(T_{x,y}) \leq B/(1-\beta).$$

2. Comme $E(\delta(X_{t+1}, Y_{t+1})) \leq \beta\delta(X_t, Y_t)$, on a
   $E(\delta(X_t, Y_t)) \leq \beta^t\delta(X_0, Y_0) \leq \beta^t B$. Mais, l'inégalité de Markov ($P(X \geq a) \leq E[X]/a$) nous donne :

$$P(\delta(X_t, Y_t) \geq 1) \leq E(\delta(X_t, Y_t)).$$

   Ainsi, pour tout $X_0, Y_0 \in \Omega$ et tout $t > 0$:

$$P(X_t \neq Y_t) = P(\delta(X_t, Y_t) > 0) = P(\delta(X_t, Y_t) \geq 1) \leq E(\delta(X_t, Y_t)) \leq \beta^t B.$$

   De ce fait, pour tout $X_0, Y_0 \in \Omega$ et tout $t > 0$:

$$P(X_t = Y_t) \geq 1 - \beta^t B.$$

   Supposons que $Y_0 \in \mathcal{L}$. Alors $Y_t \in \mathcal{L}$ (car $\mathcal{L}$ est clos), et $X_t = Y_t$ implique $X_t \in \mathcal{L}$. Ainsi, pour tout $X_0 \in \Omega$ et tout $t > 0$:

$$P(X_t \in \mathcal{L}) \geq 1 - \beta^t B.$$

   On en déduit que $P(X_t \in \mathcal{L}) \geq 1 - \varepsilon$, dès que $\beta^t B \leq \varepsilon$, i.e., $t \geq \frac{\ln(B/\varepsilon)}{\ln(1/\beta)}$.
   Ainsi $P(X_t \in \mathcal{L}) \geq 1 - \varepsilon$, dès que $t \geq \frac{\ln(B/\varepsilon)}{1-\beta}$ (car $1 - \beta \leq \ln(\frac{1}{\beta})$).

□

Un théorème similaire existe aussi lorsque $\beta = 1$, soit :

$E(\delta(X_{t+1}, Y_{t+1})) \leq \delta(X_t, Y_t)$, sous la condition que la probabilité de l'événement $(X_{t+1}, Y_{t+1}) \neq (X_t, Y_t)$ peut être minoré.

Théorème 7.16. *Soit une chaîne de Markov $\mathcal{A}$ et un ensemble ergodique $\mathcal{L}$, supposons qu'il existe un coupling (X_t, Y_t) et une fonction δ on $\Omega \times \Omega$ qui prend ses valeurs dans $\{0, 1 \cdots, B\}$ et telle que $\delta(X_t, Y_t) = 0$ ssi $X_t = Y_t$, et tel qu'il existe $\alpha > 0$ vérifiant, pour tout (X_t, Y_t) avec $X_t \neq Y_t$:*

$E(\delta(X_{t+1}, Y_{t+1})) \leq \delta(X_t, Y_t)$

$$\wedge \quad P(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq \alpha. \qquad (**)$$

Alors

$\mathcal{L}$ est l'unique ensemble ergodique $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$. De plus :

1. Le temps de hitting vérifie : $\mathbf{H}_{\mathcal{L}} \leq B^2/\alpha$.

2. Le temps d'ε-absorption vérifie :

$$\mathbf{\Theta}_{\mathcal{L}}(\varepsilon) \leq \lceil e\frac{B^2}{\alpha} \rceil \lceil \ln(\frac{1}{\varepsilon}) \rceil.$$

Démonstration. La preuve du théorème 7.16 est analogue à celle du théorème 7.14, mais repose sur la proposition suivante (conséquence des inégalités de Doob) :

Proposition 7.17. *Supposons que $D = (D_0, D_1, \cdots)$ est un processus stochastique sur $\{0, 1, \cdots, B\}$ tel que $E[D_{t+1}] \leq D_t$. De plus supposons que $P(D_{t+1} \neq D_t) \geq \alpha$ (avec $\alpha > 0$) lorsque $D_t > 0$. Alors si τ est le premier temps tel que $D_t = 0$, on a : $E[\tau] \leq B^2/\alpha$.*

Preuve du théorème 7.16.
Soit une fonction δ vérifiant les hypothèses du théorème 7.16, montrons les conclusions 1 et 2. (Le fait que $\mathcal{L}$ est l'unique ensemble ergodique, et que $\mathcal{A}$ est auto-stabilisant se déduisent de la conclusion 1, par la proposition 7.11.)

1. Considérons deux éléments x, y de Ω et un coupling (X_t, Y_t) d'état initial $(X_0, Y_0) = (x, y)$. Soit $D_t = \delta(X_t, Y_t)$ pour $t \geq 0$.
   Comme $\delta(X_t, Y_t) = 0$ ssi $X_t = Y_t$, la quantité $T_{x,y}$ est le temps nécessaire à D_t pour atteindre 0. De plus, par la proposition 7.17, on a, pour tout $x, y \in \Omega$, $E(T_{x,y}) \leq B^2/\alpha$. Et, grâce au théorème 7.13, on déduit :

   $$\mathbf{H}_{\mathcal{L}} \leq max_{x,y}\ E(T_{x,y}) \leq B^2/\alpha.$$

2. Soit $D_t = \delta(X_t, Y_t)$. Il est évident que D_t vérifie les hypothèses de la proposition 7.17. Rappelons que $T_{x,y}$ est le premier temps tel que $X_t = Y_t$, soit $D_t = 0$, lorsque $X_0 = x$ et $Y_0 = y$. On en déduit par la proposition 7.17 :

   $$E(T_{x,y}) \leq B^2/\alpha.$$

   Soit $T = \lceil eB^2/\alpha \rceil$, par l'inégalité de Markov on a la probabilité que $T_{x,y} > T$ est au plus e^{-1}. Si nous lançons s exécutions indépendantes de longueur T de la chaîne de Markov alors la probabilité pour que X_t et Y_t ne soient pas couplés à la fin de ces exécutions est au plus de e^{-s}.
   Ainsi, pour $t > T\lceil \ln(\varepsilon^{-1}) \rceil$, la probabilité de l'événement $X_t = Y_t$ est au moins $1 - \varepsilon$. Supposons que $Y_0 \in \mathcal{L}$. Alors $Y_t \in \mathcal{L}$ (car $\mathcal{L}$ est clos), et le fait que $X_t = Y_t$ implique $X_t \in \mathcal{L}$. Donc, pour tout $X_0 \in \Omega$ et tout $t > T\lceil \ln(\varepsilon^{-1}) \rceil$: $P(X_t \in \mathcal{L}) \geq 1 - \varepsilon$.

□

De cette manière, trouver un coupling (X_t, Y_t) et une fonction δ telle que $(*)$ (resp. $(**)$) nous permet de prouver que $\mathcal{A}$ est auto-stabilisant, mais aussi de calculer des bornes supérieures pour deux notions distinctes de temps de convergence.

Pour illustrer ces théorèmes, appliquons-les à l'algorithme de Herman.

Exemple 7.18. *Reprenons l'exemple de l'algorithme de Herman, et trouvons une distance sur les couples de configurations. Nous ne donnerons pas en détail l'ensemble des preuves ici, car le calcul du temps de convergence pour Herman est plus simple par la technique du path coupling que nous décrivons ci-après.*

- Coupling : *Le coupling que nous choisissons est extrêmement simple ; nous ne forçons les choix probabilistes uniquement si la machine i a une transition probabiliste à faire à la fois dans la configuration X_t et dans la configuration Y_t (i.e., lorsque $X_t(i) = X_t(i-1)$ et $Y_t(i) = Y_t(i-1)$) ; dans ce cas les machines i de X_t et Y_t sont forcés de faire le même choix probabilistes, ainsi $X_{t+1}(i)$ et $Y_{t+1}(i)$ coïncident :*

$$X_{t+1}(i) = Y_{t+1}(i) = \begin{cases} 0 & \text{avec probabilité } 1/2, \\ 1 & \text{avec probabilité } 1/2. \end{cases}$$

- Fonction δ : *la fonction $\delta(X_t, Y_t)$ est assez simple elle correspond à la distance maximale entre deux machines dont les valeurs diffèrent entre X_t et Y_t. L'évolution de δ est donnée dans la figure 7.4. Ainsi on obtient sans entrer dans les détails :*

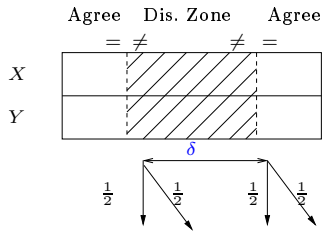


FIG. 7.4 – L'évolution de δ sous le coupling

$$\delta = \begin{cases} \delta + 1 & \text{avec probabilité } 1/4 \\ \delta - 1 & \text{avec probabilité } 1/4 \\ \delta & \text{avec probabilité } 1/2 \end{cases}$$

Donc, $E(\delta(X_{t+1}, Y_{t+1})) = \delta(X_t, Y_t)$

Et, $P(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq \frac{1}{2}$

D'après le théorème 7.16 le temps de hitting est majoré par $2N^2$, on obtient également une borne quadratique du temps d'ε-absorption.

♣

7.4.2 Comparaison des méthodes

Avant de s'intéresser à l'application de ces résultats sur des exemples, nous pouvons nous attarder sur la comparaison entre la méthode classique de preuve, et celle que nous proposons [FMP04b].
La méthode classique ne permet pas à la fois de prouver la convergence et de trouver le temps

de convergence. De plus, elle nécessite une bonne connaissance de l'algorithme (notamment de l'ensemble des états légitimes) car il faut trouver la distance (verticale) sur les configurations. Le temps de convergence, quant à lui, nécessite classiquement un calcul matriciel compliqué et faisable uniquement dans le cas où l'on peut réduire l'espace d'états. Notre méthode permet de simplifier le calcul et d'obtenir des bornes pour plusieurs notions de temps de convergence. La figure 7.5 illustre la comparaison des deux preuves.

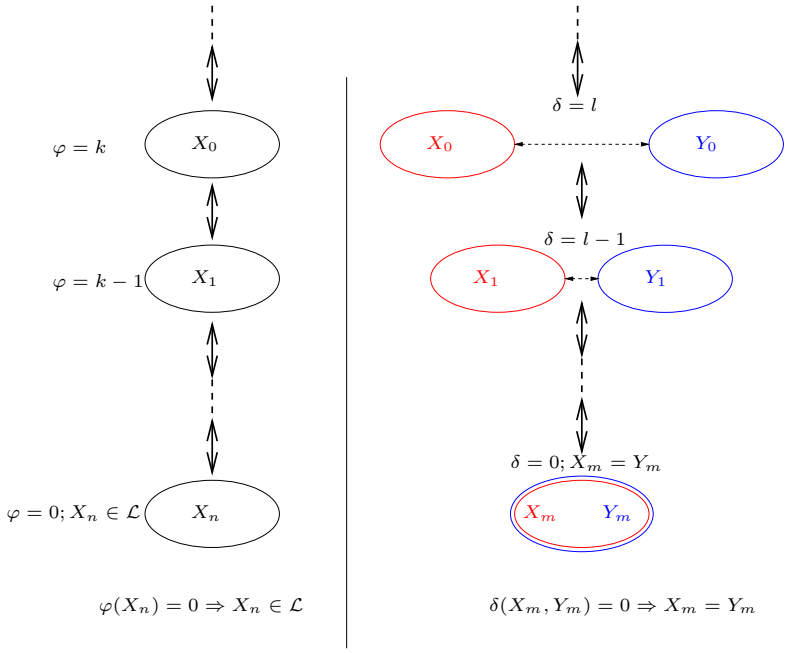


FIG. 7.5 – Une comparaison entre les preuves de convergences

7.5 Optimisation par path-coupling

Comme le fait remarquer [Ran03], il est souvent assez difficile de mesurer la variation de la distance entre deux configurations quelconques.
La méthode du *path coupling*, introduite par Bubley et Dyer [BD97], simplifie grandement les preuves, car elle permet de ne considérer que les paires *voisines*. Cette technique, comme nous le verrons à la fin de ce cette section simplifie grandement les preuves de convergence et permet dans certains cas d'obtenir de meilleures bornes que celles obtenues par les méthodes classiques [DGG+02].

7.5.1 Théorème

La méthode de path coupling nécessite de définir un coupling (X_t, Y_t) en considérant un *chemin* $X_t = Z_0, Z_1, \cdots, Z_r = Y_t$ entre X_t et Y_t dans lequel les Z_i vérifient un certain nombre de conditions.
La version suivante donné par Dyer et Greenhill de la méthode de path coupling est celle qui répondra le plus à nos attentes :

Lemme 7.19. (Dyer and Greenhill [DG98]) *Soit δ une métrique (vérifiant l'inégalité triangulaire) définie sur $\Omega\times\Omega$ et qui prend ses valeurs sur $\{0,\cdots,B\}$. Soit U un sous-ensemble de $\Omega\times\Omega$ tel que, pour tout $(X_t,Y_t)\in\Omega\times\Omega$, il existe un chemin $X_t=Z_0,Z_1,\cdots,Z_r=Y_t$ entre X_t et Y_t avec $(Z_i,Z_{i+1})\in U$ pour $0\leq i<r$ et*

$$\sum_{i=0}^{r-1}\delta(Z_i,Z_{i+1})=\delta(X_t,Y_t).$$

Supposons qu'il existe un coupling $(X,Y)\mapsto(X',Y')$ de la chaîne de Markov $\mathcal{A}$ sur tous les couples $(X,Y)\in U$, et une constante $\beta\leq 1$ telle que, pour tout $(X,Y)\in U$:

$$E[\delta(X',Y')]\leq\beta\delta(X,Y). \qquad (***)$$

Alors ce coupling peut être étendu à un coupling de $\mathcal{A}$ sur tous les couples $(X,Y)\in\Omega\times\Omega$ qui vérifie aussi $(***)$.

Deux configurations X et Y sont dites *adjacentes* si $(X,Y)\in U$. L'avantage de ce lemme est qu'il permet de vérifier la propriété cruciale $(***)$ des théorèmes précédents seulement sur l'ensemble U des couples de configurations adjacentes au lieu de sur tout l'espace $\Omega\times\Omega$. On pourra consulter les preuves de ce théorème dans [DG98].

La figure 7.6 illustre de quel manière le coupling défini sur les configurations adjacentes s'étend à l'espace tout entier.

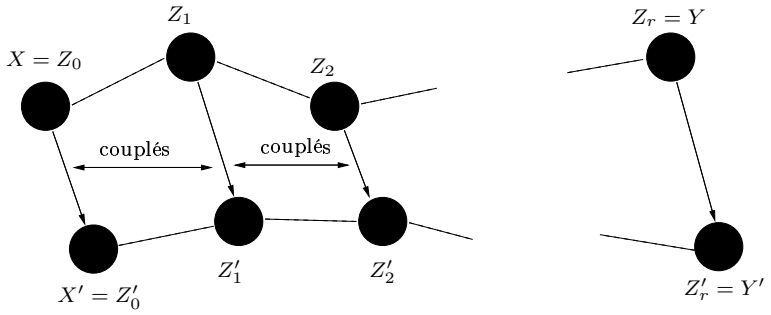


FIG. 7.6 – Etendre un coupling à partir d'un chemin dans U

Le lemme 7.19 combiné au théorème 7.14 (resp. théorème 7.16) permet d'améliorer nettement nos preuves de convergence et de vitesse de convergence pour les algorithmes auto-stabilisants.
La section suivante sera consacrée à son application à deux exemples, l'algorithme de Herman, et le problème du dilemme du prisonnier itéré.

7.5.2 Applications

Herman

Revenons à l'algorithme de Herman.

Théorème 7.20. *Pour l'algorithme de Herman et N impair, il existe un sous ensemble U de $\Omega \times \Omega$, et une métrique δ sur $\Omega \times \Omega$ prenant ses valeurs dans $\{0, \cdots, N\}$ et vérifiant les hypothèses du lemme 7.19, et un coupling défini sur U tels que :*

$$\forall (X_t, Y_t) \in U \quad E[\delta(X_{t+1}, Y_{t+1})] \leq \delta(X_t, Y_t),$$

et $\forall (X_t, Y_t) \in \Omega \times \Omega$ (*avec* $X_t \neq Y_t$) :

$$P[\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)] \geq 1/2.$$

Démonstration. – *Métrique δ.* Nous définissons δ comme la distance de Hamming, soit : $\delta(X_t, Y_t)$ est le nombre de machines sur lesquelles X_t et Y_t prennent des valeurs différentes (notons l'extrême simplicité de cette distance qui ne nécessite aucune connaissance théorique de l'algorithme).

Le couple (X_t, Y_t) appartient à U ssi $\delta(X_t, Y_t) = 1$.

– *Coupling.* Le coupling est le même que celui utilisé dans la preuve par coupling. Soit pour tout i $(1 \leq i \leq N)$, si $X_t(i)$ et $Y_t(i)$ ont des transitions probabilistes à faire (i.e., lorsque $X_t(i) = X_t(i-1)$ et $Y_t(i) = Y_t(i-1)$), les i^{me} machines de X_t et Y_t sont forcés de faire le même choix probabiliste, ainsi $X_{t+1}(i)$ et $Y_{t+1}(i)$ coïncident :

$$X_{t+1}(i) = Y_{t+1}(i) = \begin{cases} 0 & \text{avec probabilité } 1/2, \\ 1 & \text{avec probabilité } 1/2. \end{cases}$$

– *Preuve de $E[\delta(X_{t+1}, Y_{t+1})] = \delta(X_t, Y_t)$ et $P(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq 1/2$.*
A chaque étape les états des machines $1, \cdots, N$ sont mis à jours car l'algorithme est synchrone. Soit ℓ la position sur laquelle les deux configurations sont en désaccord (rappelons que $(X_t, Y_t) \in U$).
Afin de simplifier considérons (sans perte de généralité) les configurations suivantes.

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} \nu_1 & \nu_2 & \cdots & \nu_{\ell-2} & 0 & \mathbf{0} & 0 & \nu_{\ell+2} & \cdots & \nu_N \\ \nu_1 & \nu_2 & \cdots & \nu_{\ell-2} & 0 & \mathbf{1} & 0 & \nu_{\ell+2} & \cdots & \nu_N \end{pmatrix}$$

où tous les ν_i sont dans $\{0, 1\}$, les valeurs en gras correspondent à la position ℓ.

Après une étape on a :

$$\begin{pmatrix} X_{t+1} \\ Y_{t+1} \end{pmatrix} = \begin{pmatrix} \nu'_1 & \nu'_2 & \cdots & \nu'_{\ell-2} & \nu'_{\ell-1} & ? & ? & \nu'_{\ell+2} & \cdots & \nu'_N \\ \nu'_1 & \nu'_2 & \cdots & \nu'_{\ell-2} & \nu'_{\ell-1} & \mathbf{0} & 1 & \nu'_{\ell+2} & \cdots & \nu'_N \end{pmatrix}$$

'?' signifie "0 avec probabilité 1/2 et 1 avec probabilité 1/2". Notons que, pour $1 \leq i \leq \ell - 1$ et $\ell + 2 \leq i \leq N$, $X_{t+1}(i) = Y_{t+1}(i) = \nu'_i$ grâce à notre coupling.

Ainsi X_{t+1} et Y_{t+1} coïncident partout sauf, dans certains cas, aux positions ℓ ou $\ell + 1$. On a donc :

$$\delta(X_{t+1}, Y_{t+1}) = \begin{cases} 1 & \text{avec probabilité } 1/2, \\ 0 & \text{avec probabilité } 1/4, \\ 2 & \text{avec probabilité } 1/4. \end{cases}$$

De ce fait $E(\delta(X_{t+1}, Y_{t+1})) = \delta(X_t, Y_t)$, pour tout $(X_t, Y_t) \in U$. Enfin, il reste à prouver que $P(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq 1/2$, pour tout $(X_t, Y_t) \in \Omega \times \Omega$ tel que $X_t \neq Y_t$.

Notons q le nombre de jetons en désaccord (un jeton en désaccord est un site i tel que $X_t(i-1) = X_t(i) \neq Y_t(i-1) = Y_t(i)$) et notons p le nombre de zones de positions de désaccord contiguës.
Identifions les trois sources d'évolution des zones de désaccord :

1. En raison du coupling, chaque jeton en désaccord $X_t(i-1) = X_t(i) \neq Y_t(i-1) = Y_t(i)$ se transforme en une position d'accord $X_{t+1}(i) = Y_{t+1}(i)$ avec probabilité 1.

2. Chaque première position dans une zone de désaccord, i, telle que $X_t(i-1) = Y_t(i-1)$ et $X_t(i) \neq Y_t(i)$ peut se transformer en une position d'accord avec probabilité $1/2$. On notera r le nombre de tels i $(0 \leq r \leq p)$.

3. chaque première position dans une zone d'accord, i, telle que $X_t(i-1) \neq Y_t(i-1)$ et $X_t(i) = Y_t(i)$ se transforme en une position de désaccord avec probabilité $1/2$. On notera s le nombre de tels i $(0 \leq s \leq p)$.

Les cas 2 et 3 sont décrits dans la figure 7.7.

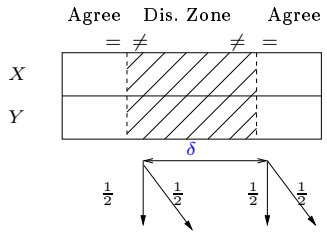


FIG. 7.7 – Evolution d'une zone de désaccord

On a : $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t) - q - r + s$.

De ce fait, l'événement $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t)$ correspond à tous les cas où $q + r = s$.

Si $q > p$, un tel événement ne peut pas se produire (probabilité 0).

Sinon, sa probabilité vaut :

$\frac{1}{4^p} \sum_{r=0}^{p-q} \binom{p}{r} \binom{p}{q+r} = \frac{1}{4^p} \sum_{r=0}^{p-q} \binom{p}{r} \binom{p}{p-q-r} \leq \frac{1}{4^p} \binom{2p}{p-q} \leq \frac{1}{2^{2p}} \binom{2p}{p}$ (par la convolution de Vandermonde [GDO94])

puis on peut majorer par $\frac{1}{2}$ (par induction sur p).

Ainsi $Pr(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq \frac{1}{2}$.

□

Comme $\delta(X_t, Y_t)$ prend ses valeurs dans $\{0, 1, \cdots, N\}$, on peut déduire des théorèmes 7.16, lemme 7.19 et théorème 7.20 :

Corollaire 7.21. *Pour N impair, l'algorithme de Herman est auto-stabilisant vers l'ensemble $\mathcal{L}$ des configurations comprenant un seul jeton, de plus :*

1. *Le temps de hitting vérifie :* $\mathbf{H}_{\mathcal{L}} \leq 2N^2$.

2. *Le temps d'ε-absorption vérifie :*

$$\mathbf{\Theta}_{\mathcal{L}}(\varepsilon) \leq 2eN^2 \lceil \ln(\frac{1}{\varepsilon}) \rceil.$$

Cette méthode peut être appliquée à plusieurs autres algorithmes auto-stabilisants sur des anneaux (par exemple, les algorithmes d'exclusion mutuelle de Flatebo et Datta [FD94] avec une politique central et aléatoire, ou l'algorithme d'horloge binaire de Mayer,Ostrovsky et Yung sous une politique synchrone [MOY96]).

Dilemme du prisonnier itéré

Dans cette dernière section nous allons nous intéresser au problème du prisonnier itéré. En voici une description :

Exemple 7.22. *Nous allons considérer le problème du dilemme du prisonnier itéré comme il est modélisé dans [DGG+02].*
Ce problème issu de la théorie des jeux entre dans la classe dite des problèmes d'apprentissage multi-agents. Le dilemme du prisonnier itéré est un modèle de jeu coopératif à N joueurs. Il permet d'illustrer un mécanisme qui assure l'émergence de la coopération entre les agents sans un contrôle centralisé et qui obéissent tous à des règles d'apprentissage similaires. Le problème du calcul du temps de convergence (temps de coopération) est donc l'enjeu principal dans l'analyse de ce processus.
Si on considère le dilemme du prisonnier à deux joueurs [Axe84], chaque joueur peut choisir de coopérer ou de dénoncer. Si les deux joueurs coopèrent, ils reçoivent une récompense de R points. Si les deux joueurs se dénoncent ils touchent seulement P points. Enfin, si un seul joueur coopère il reçoit S points tandis que son adversaire reçoit T points. Les paramètres vérifient $T > R > P > S$ et $2R > T + S$. Ainsi pour un jeu à un seul tour on peut remarquer qu'il vaut mieux dénoncer son adversaire même si ce n'est pas complètement optimal. Dans le problème du dilemme du prisonnier itéré, on joue plusieurs tours et les joueurs peuvent baser leurs décisions sur les résultats des tours précédents. Une preuve empirique [Axe84] donne une stratégie efficace pour ce jeu, la stratégie de Pavlov : si un joueur est récompensé par T ou R points durant un tour, alors il répète son choix précédent, sinon il change. On peut remarquer

que la stratégie de Pavlov peut aussi être décrite ainsi : un joueur coopère uniquement s'il a fait le même choix que son adversaire au tour précédent. C'est cette stratégie que nous modélisons.

Passons à présent à sa description formelle :

La topologie est un anneau bidirectionnel de N sommets, et la politique est aléatoire centralisé (on tire au sort de manière uniforme les machines qui seront activées). L'ensemble des états possibles pour les machines est $Q = \{-, +\}$.

A chaque étape, une machine i ($1 \leq i \leq N$) est choisi aléatoirement de manière uniforme, et les valeurs $x(i)$ et $x(i+1)$ deviennent $x'(i)$ et $x'(i+1)$ respectivement de la manière suivante :

- *si $x(i) = x(i+1)$, alors $x'(i) = x'(i+1) = +$,*
- *si $x(i) = \neg x(i+1)$, alors $x'(i) = x'(i+1) = -$.*

(Lorsque $i = N$, $(i+1)$ vaut ici 1. De plus, $\neg +$ représente $-$, et $\neg -$, $+$.) ♣

Notons que dans cet exemple $\mathcal{L}$ est l'ensemble comprenant une unique configuration x^*, avec $x^*(i) = +$ pour tout $1 \leq i \leq N$, et est ergodique.

Montrons à présent que l'algorithme est auto-stabilisant.

Théorème 7.23. *Pour l'algorithme du dilemme du prisonnier itéré , il existe un sous-ensemble U de $\Omega \times \Omega$, et une métrique δ vérifiant les hypothèses du lemme 7.19 sur $\Omega \times \Omega$ prenant ses valeurs sur $\{0, \cdots, 11N\}$, et un coupling défini sur U tel que, pour tout $(X_t, Y_t) \in U$:*

$$E[\delta(X_{t+1}, Y_{t+1})] \leq (1 - \frac{1}{18N})\delta(X_t, Y_t).$$

Démonstration. – *Configurations adjacentes.* Un couple (X, Y) appartient à U ssi X et Y coïncident partout sauf sur k positions contiguës, avec $k = 1, 2, 3, 4$ ou 5, sur lesquelles elles peuvent ne pas coïncider .

– *Fonction δ.* Considérons un couple $(X, Y) \in U$ tel que X et Y sont en désaccord sur k positions contiguës avec $(1 \leq k \leq 5)$.
Soit $\delta(X, Y) = a_k$ où a_k est une constante positive qui sera déterminée plus tard. Par convention, on pose $a_0 = 0$.
La fonction δ sur U s'étend à l'espace tout entier $\Omega \times \Omega$ de la manière suivante.
Soit $(X, Y) \in \Omega \times \Omega$ tel que X et Y ne différent que sur ℓ positions contiguës. On a : $\ell = 5m + r$ pour un $m \geq 0$ et $0 \leq r \leq 4$. la fonction δ est ainsi définie par : $\delta(X, Y) = ma_5 + a_r$.

Supposons que X et Y coïncident sauf sur n zones disjointes de positions contiguës W_p $(1 \leq p \leq n)$. Soit m_p et r_p le quotient et le reste de la division euclidienne de la longueur de W_p par 5 ($|W_p| = 5m_p + r_p$ avec $0 \leq r_p \leq 4$).

Alors, pour tout $(X, Y) \in \Omega \times \Omega$, δ est définie par : $\delta(X, Y) = \sum_{p=1}^{n} m_p a_5 + a_{r_p}$.

Nous allons montré, à présent, que pour des valeurs bien choisies de a_k ($1 \leq k \leq 5$), la fonction δ est une métrique qui vérifie les conditions requises par le lemme 7.19 (il existe un chemin $X = Z_0, Z_1, \cdots, Z_r = Y$ où $(Z_j, Z_{j+1}) \in U$, $\sum_{j=0}^{r-1} \delta(Z_j, Z_{j+1}) = \delta(X, Y)$).

– *Coupling.* Le coupling $(X, Y) \mapsto (X', Y')$ est défini tel que, à chaque étape, on choisisse la même position pour la chaîne de Markov X et pour Y. Ainsi, à chaque étape, l'état de la machine sélectionnée (disons j) et l'état de la machine $j+1$ sont mis à jour simultanément dans X et dans Y.

– *Preuve de* $E[\delta(X', Y')] \leq \beta\delta(X, Y)$. Considérons un couple $(X_t, Y_t) \in U$ avec k positions contiguës de désaccord. Soit i la première position de désaccord. Le vecteur

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix}$$

est de la forme

$$\begin{pmatrix} \gamma_1 & \cdots & \gamma_{i-2} & \gamma_{i-1} & \gamma_i & \cdots & \gamma_{i+k-1} & \gamma_{i+k} & \cdots & \gamma_N \\ \gamma_1 & \cdots & \gamma_{i-2} & \gamma_{i-1} & \neg\gamma_i & \cdots & \neg\gamma_{i+k-1} & \gamma_{i+k} & \cdots & \gamma_N \end{pmatrix}$$

où les γ_ℓ sont dans $\{-, +\}$. Supposons que la machine sélectionnée aléatoirement j est telle que $1 \leq j \leq i-2$ ou $i+k \leq j \leq N$. Alors $X_t(j) = Y_t(j)$ et $X_t(j+1) = Y_t(j+1)$, et donc $X_{t+1}(j) = Y_{t+1}(j)$ et $X_{t+1}(j+1) = Y_{t+1}(j+1)$; la zone de désaccord n'est donc pas modifiée.
Supposons à présent que la machine j est la machine $i-1$. Alors, après une étape, le vecteur :

$$\begin{pmatrix} X_{t+1} \\ Y_{t+1} \end{pmatrix}$$

est de la forme

$$\begin{pmatrix} \gamma_1 & \cdots & \gamma_{i-2} & \gamma'_{i-1} & \gamma'_i & \cdots & \gamma_{i+k-1} & \gamma_{i+k} & \cdots & \gamma_N \\ \gamma_1 & \cdots & \gamma_{i-2} & \neg\gamma'_{i-1} & \neg\gamma'_i & \cdots & \neg\gamma_{i+k-1} & \gamma_{i+k} & \cdots & \gamma_N \end{pmatrix}$$

où $\gamma'_{i-1} = \gamma'_i = +$ si $\gamma_{i-1} = \gamma_i$, et $\gamma'_{i-1} = \gamma'_i = -$ sinon.

Cela implique que la zone de désaccord à progressé d'une position vers la gauche. Un cas symétrique apparaît lorsque $j = i+k-1$. Nous dirons que j est une "position extérieure". Une simple analyse de cas que nous donnons ci-après montre que il existe β pour des valeurs appropriés de a_k ($1 \leq k \leq 5$), avec $\beta \leq 1 - \frac{1}{18N}$ tel que $E[\delta(X', Y')] \leq \beta\delta(X, Y)$. De plus pour ces valeurs de a_k, la valeur maximale B de δ sur $\Omega \times \Omega$ est telle que $B \leq 11N$.
Montrons ce résultat en détail ce qui achèvera la preuve.

Considérons $(X, Y) \in U$. Soit $[i, i+k-1]$ l'intervalle des positions contiguës de désaccord dans les configurations X et Y (avec $1 \leq k \leq 5$). Le choix aléatoire de la machine sélectionnée j modifie la zone de désaccord ssi j correspond à un des cas suivants :

- *Position extérieure :* Ceci signifie que $j = i - 1$ ou $j = i + k - 1$. Il y a deux positions extérieures pour tout $1 \leq k \leq 5$. Choisir une position extérieure fait grandir la zone de désaccord d'une position . Cela arrive avec la probabilité $2/N$ et cela modifie $E(\delta)$ de : $\frac{2}{N}(a_{k+1} - a_k)$. (Pour $k = 5$, $a_{k+1} = a_6$ représente $a_5 + a_1$.)

- *Position limite :* Ceci signifie que $j = i$ ou $j = i + k - 2$. Il n'y a pas de position limite si $k = 1$ (elle est confondue avec une des deux positions extérieures), une position limite si $k = 2$, et deux positions limites si $k = 3, 4, 5$.
  Le fait de choisir une position limite fait décroître la taille de la zone de désaccord de 2.
  Cela se produit avec probabilité $1/N$ (resp. $2/N$) lorsque $k = 2$ (resp. $k = 3, 4, 5$).
  Cela contribue à la modification de $E(\delta)$ de $\frac{1}{N}(a_0 - a_2) = -\frac{1}{N}a_2$ quand $k = 2$, et de $\frac{2}{N}(a_{k-2} - a_k)$ quand $k = 3, 4, 5$.

- *Position intérieure :* Ceci signifie que $j = i+1$ ou $j = i+k-3$. Il n'y a pas de position intérieure si $k = 1, 2$ ou 3, une seule si $k = 4$, et deux si $k = 5$.
  Pour $k = 4$, Choisir une position intérieure ($j = i+1$) transforme la zone de désaccord en deux zones de désaccord de longueur 1. Cela se produit avec la probabilité $1/N$, et contribue à la modification de $E(\delta)$ de : $\frac{1}{N}(2a_1 - a_4)$.

  Pour $k = 5$, Le fait de choisir une position intérieure ($j = i+1$ or $j = i+2$) transforme la zone de désaccord en deux zones de désaccord de longueur 1 et 2. Cela se produit avec probabilité $2/N$, et contribue à la modification de $E(\delta)$ de : $\frac{2}{N}(a_1 + a_2 - a_5)$.

De ce fait, nous avons les cas suivants :

1. *Cas k=1.* Alors :
   $E(\delta(X', Y')) - \delta(X, Y) = \frac{2}{N}(a_2 - a_1)$.
   Ainsi $E(\delta(X', Y')) = \beta_1 \delta(X, Y)$ avec $\beta_1 = 1 - \frac{2}{N}\frac{a_1 - a_2}{a_1}$
   (en utilisant le fait que $\delta(X, Y)$ est ici égal à a_1).

2. *Cas k=2.* Alors :
   $E(\delta(X', Y')) - \delta(X, Y) = \frac{1}{N}(2(a_3 - a_2) + (a_0 - a_2)) = \frac{1}{N}(2a_3 - 3a_2)$.
   Ainsi $E(\delta(X', Y')) = \beta_2 \delta(X, Y)$ avec $\beta_2 = 1 - \frac{1}{N}\frac{3a_2 - 2a_3}{a_2}$
   (en utilisant le fait que $\delta(X, Y)$ est ici égal à a_2).

3. *Cas k=3.* Alors :
   $E(\delta(X', Y')) - \delta(X, Y) = \frac{2}{N}((a_4 - a_3) + (a_1 - a_3)) = \frac{2}{N}(a_4 - 2a_3 + a_1)$.
   Ainsi $E(\delta(X', Y')) = \beta_3 \delta(X, Y)$ avec $\beta_3 = 1 - \frac{2}{N}\frac{2a_3 - a_4 - a_1}{a_3}$
   (en utilisant le fait que $\delta(X, Y)$ est ici égal à a_3).

4. *Cas k=4.* Alors :
   $E(\delta(X', Y')) - \delta(X, Y) = \frac{1}{N}(2(a_5 - a_4) + 2(a_2 - a_4) + (2a_1 - a_4)) = \frac{1}{N}(2a_5 - 5a_4 + 2a_2 + 2a_1)$.
   Ainsi $E(\delta(X', Y')) = \beta_4 \delta(X, Y)$ avec $\beta_4 = 1 - \frac{1}{N}\frac{5a_4 - 2a_5 - 2a_2 - 2a_1}{a_4}$

(en utilisant le fait que $\delta(X,Y)$ est ici égal à a_4).

5. *Cas k=5.* Alors :
$E(\delta(X',Y')) - \delta(X,Y) = \frac{2}{N}((a_5 + a_1 - a_5) + (a_3 - a_5) + (a_1 + a_2 - a_5)) = \frac{2}{N}(-2a_5 + a_3 + a_2 + 2a_1)$.
Ainsi $E(\delta(X',Y')) = \beta_5\delta(X,Y)$ avec $\beta_5 = 1 - \frac{2}{N}\frac{2a_5 - a_3 - a_2 - 2a_1}{a_5}$
(en utilisant le fait que $\delta(X,Y)$ est ici égal à a_5).

De ce fait, pour tout $(X,Y) \in U$, $E(\delta(X',Y')) \leq \beta\delta(X,Y)$, avec $\beta = max\{\beta_k\}_{1\leq k\leq 5}$.
Il suffit à présent de trouver $a_1, \cdots, a_5$ tel que β satisfait $0 < \beta < 1$.
Une solution possible est : $a_1 = 21, a_2 = 20, a_3 = 29, a_4 = 36, a_5 = 48$.
On en déduit $\beta \leq 1 - (1/18N)$, donc $\frac{1}{1-\beta} \leq 18N$.

Montrons à présent que δ est une métrique sur $\Omega \times \Omega$, soit à montrer :

1. $\delta(X,Y) = 0$ ssi $X = Y$.

2. $\forall x, y, z\ \delta(x,z) \leq \delta(x,y) + \delta(y,z)$.

La première assertion vient du fait que tous les coefficients a_i sont strictement positifs.
La preuve de la seconde assertion repose sur le fait suivant :
Pour tout $k = 1,2,3,4,5$ et toute partition $i_1, .., i_\ell$ de k (i.e : $i_1 + .. + i_\ell = k$), on a :
$a_k < a_{i_1} + a_{i_2} + ... + a_{i_\ell}$.
Finalement, il reste à vérifier que pour tout $x, y \in \Omega \times \Omega$, il existe un chemin $x = z_0, z_1, \cdots, z_r = y$ où $(z_j, z_{j+1}) \in U$, $\sum_{j=0}^{r-1}\delta(z_j, z_{j+1}) = \delta(x,y)$. Pour tout x, y dans Ω, on considère le chemin de x à y suivant :

On élimine tout d'abord toutes les zones de désaccord de longueur 5 (en commençant par la gauche), puis par celles de longueur 4, 3, 2 et finalement 1. De par la définition de notre métrique on obtient que pour $x = z_0, z_1, \cdots, z_k = y$,on a :

$\sum_{j=0}^{k-1}\delta(z_j, z_{j+1}) = \delta(x,y)$.

Enfin, notons que la valeur maximale B de δ sur $\Omega \times \Omega$ est au plus $a_1\lceil\frac{N}{2}\rceil \leq 11N$.

□

De ce fait, la conjonction du théorème 7.14, du lemme 7.19 et du théorème 7.23 nous permet d'obtenir :

Corollaire 7.24. *L'algorithme du dilemme du prisonnier itéré est auto-stabilisant vers* $\mathcal{L} = \{(+)^N\}$. *De plus :*

1. *Le temps de hitting vérifie :* $\mathbf{H}_{\mathcal{L}} \leq 198N^2$.
2. *Le temps d'*ε*-absorption vérifie :*

$$\mathbf{\Theta}_{\mathcal{L}}(\varepsilon) \leq 18N \ln(\frac{11N}{\varepsilon}).$$

Ainsi nous avons trouvé la borne quasi-linéaire comme dans [DGG+02] pour le temps de ε-absorption. On peut remarquer que notre constante est meilleure, et que notre preuve nous donne aussi une majoration du temps de hitting pour cet algorithme qui avait été obtenu empiriquement dans [Kit95].

La méthode que nous avons choisie est plus simple que celle choisie dans [DGG+02]. La preuve de la décroissance de δ est bien plus simple à obtenir dans notre cas grâce au path coupling.

Chapitre 8

Application des techniques de physique statistique

La statistique est la première des sciences inexactes.
Edmond et Jules de Goncourt

Le coupling est une technique couramment utilisé en Physique statistique. Cela nous a donc poussé à continuer l'analogie entre les systèmes distribués probabilistes et la Physique statistique. Un grand nombre de modèles combinatoires, comme par exemple ceux de l'échantillonnage peuvent se résoudre à l'aide de la comparaison avec la Physique statistique. En effet, il existe de nombreux travaux sur les connections entre la physique statistique et l'informatique théorique [MK00, CDMM02]. Ces corrélations sont encore plus importantes dans le cas des problèmes probabilistes, car les physiciens statisticiens ont été confrontés à ces problématiques bien avant les informaticiens. A l'instar de Randall dans [Ran03], nous pouvons donner un tableau (réduit) (voir tableau 8.1) de correspondance de notions entre les problèmes de Physique statistique et ceux d'informatique.

Physique statistique	Informatique théorique
Recouvrement par des monomères et dimères	matchings
Recouvrement par des dimères	matchings parfaits
hard core lattice gas model	Ensembles indépendants
Spin	bit
Mélange polynomial	Mélange rapide

FIG. 8.1 – Quelques comparaisons entre des notions de Physique statistique et d'informatique

8.1 Quelques exemples courants

Dans cette partie, nous nous intéresserons à donner une liste (non exhaustive) d'exemples dans lesquels l'interaction avec la Physique statistique a permis de trouver des techniques de preuves et de calculs intéressantes. Ces modèles sont tirés de [Ran03].

8.1.1 Ensembles indépendants

Pour la définition de ce problème, on pourra se reporter au chapitre 6.

Soit $G = (V, E)$ un graphe et Ω l'ensemble des ensembles indépendants de G . On peut modéliser chacun des ensembles indépendants comme une fonction de V dans $\{0, 1\}$, où $f(v) = 1$ ssi v est dans l'ensemble indépendant. Ainsi, on peut définir des poids X_0 et X_1 pour contrôler l'envie d'avoir un noeud à l'intérieur ou à l'extérieur de l'ensemble. On définit alors un poids global :

$$\omega(I) = \prod_{v \in V} X_{f(v)}.$$

Notons que lorsque $X_0 = 1$ et X_1 un entier, ceci correspond à avoir X_1 particules dans chaque sommet de l'ensemble indépendant. Si nous essayons d'échantillonner des ensembles indépendants suivant cette fonction de poids, nous avons la mesure de probabilité suivante sur Ω :

$$\pi(I) = \frac{\omega(I)}{\sum_{I' \in \Omega} \omega(I')}.$$

Si on pose $\lambda = X_1$, on trouve :

$$\pi(I) = \frac{\lambda^{|I|}}{\sum_{I' \in \Omega} \lambda^{|I'|}}.$$

Lorsque $\lambda \in \mathbb{R}^+$ est grand, on favorise les ensembles indépendants denses, sinon on favorise les dispersés.

8.1.2 Coloriages

Un k-coloriage d'un graphe G est un coloriage des sommets avec k couleurs différentes tel que 2 sommets reliés entre eux par une arête ne sont pas coloriés de la même couleur.
Nous noterons, de la même façon l'ensemble des k-coloriages d'un graphe G par une fonction de V dans $\{1, .., k\}$. On peut, ici aussi, associer à chaque couleur un poids $X_0, ..., X_k$, et ainsi définir un poids global pour tout $C \in \Omega$ (Ω étant l'ensemble des k-coloriages de G) :

$$\omega(C) = \prod_{v \in V} X_{f(v)} = \prod_{i=1}^{k} X_i^{c_i}.$$

où c_i est le nombre de sommets coloriés avec la couleur i. Lorsque $X_i = 1$, le fait d'échantillonner suivant ce poids nous donne la distribution uniforme sur les k-coloriages.

8.1.3 Matchings

Un matching est un recouvrement des sommets d'un graphe par des arêtes, tel que deux arêtes distinctes ne recouvrent pas le même sommet.
Ici la fonction est de la forme $f : E \to \{0, 1\}$ et on note Ω l'ensemble des matchings de G. Comme précédemment on notera X_0 et X_1 des poids. Ainsi pour tout $M \in \Omega$, on a le poids suivant :

$$\omega(M) = \prod_{e \in E} X_{f(e)} = \mu^{|M|},$$

dans le cas où $X_0 = 1$ et $X_1 = \mu$. On peut donc échantillonner suivant ce poids.

Il existe d'autres nombreux types de modèles qui peuvent être étudiés grâce à des techniques de Physique statistique, comme les modèles d'interaction entre particules.
Décrivons à présent quelle est la technique classique pour transposer ces problèmes en des problèmes de Physique statistique.

8.1.4 Transposition avec la Physique statistique

En Physique statistique, les systèmes tendent à favoriser les configurations d'énergies minimales, et cette préférence est contrôlée par la température. La fonction d'énergie sur les configurations est déterminée par l'hamiltonien $H(\sigma)$. Etant donné que l'on cherche à minimiser l'énergie, les configurations ont un poids

$$\omega(\sigma) = e^{-\beta H(\sigma)},$$

où $\beta = 1/T$ est l'inverse de la température. La probabilité de chaque configuration est donc de $\pi(\sigma) = \omega(\sigma)/Z$, où Z est la constante normalisatrice bien connu sous le nom de fonction de partition.

De ce fait, pour notre exemple concernant les ensembles indépendants, il suffit de définir l'hamiltonien comme :

$$H(I) = -\sum_{v \in V} \delta_{v \in I}$$

où δ représente le symbole de Kronecker.

On pourrait étendre cette technique pour tous les ensembles évoqués et même pour ceux des interactions de couples en utilisant les modèles d'Ising (ferromagnétique et antiferromagnétique).

8.1.5 3 SAT aléatoire

Un autre parallèle entre Physique statistique et informatique à été mis en lumière dans [CDMM02], il concerne les grandes similitudes entre le problème 3-SAT aléatoire et les verres de spin. Nous ne détaillerons pas ici la comparaison (voir [CDMM02]).
Par contre nous pouvons rappeler que le problème 3-SAT aléatoire consiste en le tirage aléatoire d'un système de k clauses propositionnelles de taille 3 exprimées à l'aide de α variables propositionnelles. Le résultat principal consiste en l'exhibition d'un phénomène de transition de phase, en effet en fonction du rapport k/α le système tiré passe d'une très forte chance d'avoir une solution, à une chance pratiquement nulle.
Les phénomènes de transitions de phases étant courants en Physique statistique, de nombreux physiciens ont donc essayé d'analyser ce type de systèmes informatiques [CDMM02, MK00].

Toutes ces lectures ont été notre inspiration pour se pencher sur les rapports entre la Physique statistique (notamment la transition de phase ou son absence) et l'algorithmique distribuée (notamment à travers des problèmes d'auto-stabilisation).

8.2 Algorithmes distribués vus comme des champs de Markov

Dans cette partie, nous supposons que la topologie du réseau est un anneau de taille N. Nous allons utiliser ici une notion classique en Physique statistique, celle de champ de Markov (notion équivalente à celle de champ de Gibbs) ; informellement un champ de Markov est l'extension "dans l'espace" de la notion de chaîne de Markov : en effet la valeur d'un certain site ne dépend que de la valeur des sites voisins (dans le cas d'une chaîne de Markov les sites voisins sont le futur et le passé immédiat). Enfin, on peut remarquer que la notion de champs de Markov pour les traces d'exécutions est facilement généralisable dans le cas d'autres topologies.
Pour plus de détails sur les champs de Markov, on peut se reporter à [Bré99] ou à [FMP03].

8.2.1 Champs aléatoires

Une *$\mathcal{A}$-trace* (d'exécution) $\mathcal{T}$ de longueur m partant d'une configuration initiale T_0, est une suite de configurations $T_0, T_1, \cdots, T_m$ dans Q^N telle que $T_{l-1} \xrightarrow{\mathcal{A}} T_l$ pour tout $1 \leq l \leq m$.

Une telle trace peut être représentée comme une grille de taille $(m+1) \times N$, où la ligne 0 correspond à la configuration initiale T_0, et la coordonnée (l, i) $(1 \leq l \leq m,\ 1 \leq i \leq N)$ correspond à l'état de la $i^{\text{ème}}$ machine après l étapes de l'algorithme.

La probabilité associée à $\mathcal{T}$ est $\prod_{l=1}^{m} p(T_{l-1} \xrightarrow{\mathcal{A}} T_l)$.
Cela représente la probabilité, étant donnée une configuration initiale T_0, que $\mathcal{A}$ génère une suite de configurations $T_1, \cdots, T_m$ après m étapes.

Plus formellement, soit $S = \{0, 1, \cdots, m\} \times \{1, \cdots, N\}$ l'ensemble des coordonnées (ou *sites*) de $\mathcal{T}$. Soit S_l le sous-ensemble $\{(l, 1), \cdots, (l, N)\}$ de S, pour tout $0 \leq l \leq m$. Soit $S'' = S_1 \cup \cdots \cup S_{m-1}$ et $S' = S'' \cup S_m$ (ainsi : $S = S_0 \cup S' = S_0 \cup S'' \cup S_m$). On a :

Définition 8.1. *Etant donnés une chaîne de Markov $\{x_t\}_{t \geq 0}$ (associée à $\mathcal{A}$ (voir 8.2) partant d'une configuration donnée $x_0 = T_0$, et un entier $m \geq 0$, le* champ aléatoire *X des traces de $\mathcal{A}$ de longueur m partant de T_0 est une collection $X = \{X(l, i)\}_{(l,i) \in S}$ de variables aléatoires à valeur dans Q telle que, pour tout $(l, i) \in S$: $X(l, i) = x_l(i)$.*

Soit Ω_S l'ensemble de toutes les fonctions de S dans Q ; une trace peut être considérée comme un élément Ω_S, et être notée $X = \{X(s)\}_{s \in S}$ si cela concerne le champ aléatoire, et par $\mathcal{T}, \mathcal{U}, \cdots$ si nous considérons une trace fixe.

Nous utiliserons aussi ces notations pour les éléments de Ω_B, où B est un sous-ensemble de S. Pour $C \subset B \subset S$, $\eta \in \Omega_B$ et $\eta' \in \Omega_C$, on écrit "$\eta \equiv \eta'$ sur C" si la restriction de η à l'ensemble des sites de C coïncide avec η'. Pour $B \subset S$, $s \in S$ et $\eta \in \Omega_B$, l' expression $\eta(s)$ représente la valeur de η au site s, et $\eta(B)$ est une notation simplifiée pour $\{\eta(s)\}_{s \in B}$.

Définissons à présent, pour chaque $T_0 \in \Omega_{S_0}$, la distribution de probabilité conditionnelle sur $\Omega_{S'}$:

$\mathcal{P}^{\mathcal{A}}_{S'}(.,T_0) = Pr(X \equiv . \text{ sur } S' \mid X \equiv T_0 \text{ sur } S_0)$.
On a, pour tout $\mathcal{T}' \in \Omega_{S'}$:

$$\mathcal{P}^{\mathcal{A}}_{S'}(\mathcal{T}', T_0) = p(T_0 \xrightarrow{\mathcal{A}} T_1)p(T_1 \xrightarrow{\mathcal{A}} T_2)\cdots p(T_{m-1} \xrightarrow{\mathcal{A}} T_m),$$

où T_l est la restriction de $\mathcal{T}'$ sur S_l $(1 \leq l \leq m)$.

Exemple 8.2. *Considérons par exemple l'algorithme de Herman (voir 7). Supposons que* $N = 5$ *et* $m = 3$. *La grille (ou matrice) suivante*

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

représente la trace $\mathcal{T} = T_0T_1T_2T_3$ *dans laquelle* T_l *correspond à la ligne* l *de la grille (*$0 \leq l \leq 3$*).*

La probabilité d'aller de la ligne 0 *à la ligne* 1 *(resp. de* 1 *à* 2*) est* $(1/2)^3$*, (en effet il y a trois jetons).*
La probabilité d'aller de la ligne 2 *à* 3 *est de* $1/2$ *(il n'a qu'un jeton).*
Ainsi $\mathcal{P}^{\mathcal{A}}_{S'}(\mathcal{T}', T_0) = (1/2)^3(1/2)^3 1/2 = 2^{-7}$ *où* $\mathcal{T}'$ *est* $T_1T_2T_3$.

♣

Pour $T_m \in \Omega_{S_m}$, soit :

$$\mathcal{P}^{\mathcal{A}}_{S',S_m}(T_m, T_0) = \sum_{\mathcal{T}' \in \Omega_{S'}:\ \mathcal{T}' \equiv T_m \text{ on } S_m} \mathcal{P}^{\mathcal{A}}_{S'}(\mathcal{T}', T_0).$$

$\mathcal{P}^{\mathcal{A}}_{S',S_m}(.,T_0)$ est appelée la *distribution marginale* sur S_m de $\mathcal{P}^{\mathcal{A}}_{S'}(.,T_0)$.

Proposition 8.3. *On a, pour tout* $T_0, T_m \in Q^N$ *:*

$$\mathcal{P}^{\mathcal{A}}_{S',S_m}(T_m, T_0) = Pr(x^m = T_m \mid x^0 = T_0).$$

Informellement, la distribution marginale sur S_m de la distribution de probabilité de X est égale à la distribution de x_m.

Démonstration.

$$\mathcal{P}^{\mathcal{A}}_{S',S_m}(T_m, T_0) = \sum_{\mathcal{T}' \in \Omega_{S'}:\ \mathcal{T}' \equiv T_m \text{ sur } S_m} \mathcal{P}^{\mathcal{A}}_{S'}(\mathcal{T}', T_0)$$

$$= \sum_{U_1 \in Q^N, \cdots, U_{m-1} \in Q^N} Pr(X \equiv (U_1U_2\cdots U_{m-1}T_m) \text{ sur } S' \mid X \equiv T_0 \text{ sur } S_0)$$

$$= \sum_{U_1 \in Q^N, \cdots, U_{m-1} \in Q^N} p(T_0 \xrightarrow{\mathcal{A}} U_1)p(U_1 \xrightarrow{\mathcal{A}} U_2)\cdots p(U_{m-1} \xrightarrow{\mathcal{A}} T_m)$$

$$= Pr(x^m = T_m \mid x^0 = T_0).$$

□

Définition 8.4. *Soit T_0, U_0 des configurations de Q^N, et $\{(x_t, y_t)\}_{t\geq 0}$ un coupling (voir le chapitre 6 pour les définitions) de $\mathcal{A}$ partant de (T_0, U_0). Le* coupling induit de traces de $\mathcal{A}$ de longueur m *est la variable aléatoire (X, Y) à valeur dans $\Omega_S \times \Omega_S$ telle que, pour tout $(l, i) \in S$, on ait :*

$X(l,i) = x_l(i)$ *et* $Y(l,i) = y_l(i)$.

On dira qu'un coupling $\{(x_t, y_t)\}_{t\geq 0}$ est *localement stable* ssi, pour tout $1 \leq i \leq N$, et pour tout $t \geq 0$:
$(x_t(i-1) = y_t(i-1) \wedge x_t(i) = y_t(i)) \Rightarrow x_{t+1}(i) = y_{t+1}(i)$.

Nous utiliserons $(x, y)_t(i)$ pour abréviation de $(x_t(i), y_t(i))$.
Avec cette notation, la stabilité locale peut se décrire de la manière suivante, pour tout $q, r, s \in Q$, tout $1 \leq i \leq N$ et tout $t \geq 0$:

$((x, y)_t(i-1) = (q, q) \wedge (x, y)_t(i) = (r, r)) \Rightarrow (x, y)_{t+1}(i) = (s, s)$.

On peut remarquer que si le coupling $\{(x_t, y_t)\}_{t\geq 0}$ est localement stable, alors le coupling induit (X, Y) est tel que, pour tout $(l, i) \in S'$:

$(X(l-1, i-1) = Y(l-1, i-1) \wedge X(l-1, i) = Y(l-1, i)) \Rightarrow X(l, i) = Y(l, i)$.

8.2.2 Voisinage

Pour un site $s \in S'$, le *voisinage* est défini comme suit :

- Pour $s = (l, i) \in S''$ le voisinage de s, noté $\mathcal{N}_s$, est le 6-uplet de sites $\langle (l-1, i-1), (l-1, i), (l, i-1), (l, i+1), (l+1, i), (l+1, i+1) \rangle$. (voir figure 8.2).

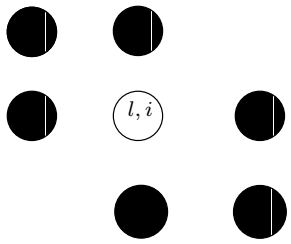


FIG. 8.2 – Voisinage pour $s \in S''$.

- Pour $s = (m, i) \in S_m$, le voisinage de s, noté $\mathcal{O}_s$, est le couple de sites $\langle (m-1, i-1), (m-1, i) \rangle$. (voir figure 8.3).

Nous dirons que de sites r et s de S (non tous les deux dans S_0) sont *voisins* ou *adjacents*, et nous écrirons $r \sim s$, si :

- $s \in \mathcal{N}_r$ si $r \in S''$, ou
- $s \in \mathcal{O}_r$ si $r \in S_m$,

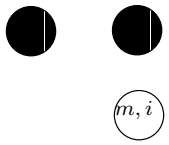


FIG. 8.3 – Voisinage pour $s \in S_m$.

ou symétriquement en inversant r et s.

Pour $m = 3$ et $N = 5$, le graphe des sites adjacents (l, i), avec $0 \leq l \leq m = 3$ et $1 \leq i \leq N = 5$, est représenté en figure 8.4. (Pour la simplicité les arêtes entre les sites $(l, 1)$ et les sites $(l, 5), (l-1, 5)$, avec $l = 1, 2, 3$, n'ont pas été dessinées).

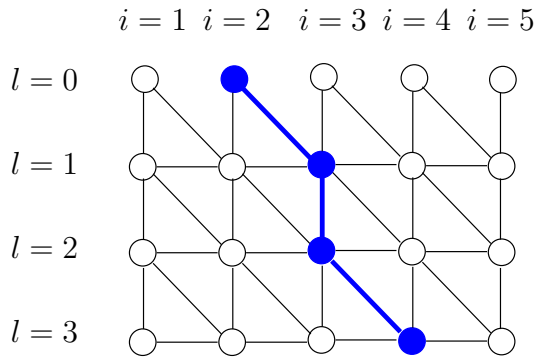


FIG. 8.4 – Graphe des sites adjacents.

Par un *chemin* de S_i à S_j, nous entendons une suite de sites distincts $s_1, s_2, \cdots, s_k$ dans laquelle les sites consécutifs sont adjacents (i.e. : $s_1 \sim s_2 \sim \cdots \sim s_k$), et avec $s_1 \in S_i$ et $s_k \in S_j$.

Un *chemin S-SE* de S_i à S_j est un chemin $s_1 \sim s_2 \sim \cdots \sim s_k$ de S_i à S_j qui ne prend des transitions que dans les directions "Sud" ou "Sud-Est" ; soit telles que, pour tout h $(1 \leq h \leq k-1)$, si le site s_h est (l, i), alors s_{h+1} est $(l+1, i)$ (vers le Sud) ou $(l+1, i+1)$ (vers le Sud-Est). Un exemple de chemin S-SE est dessiné en gras dans la figure 8.4.

Il est facile de voir que le champ des traces de $\mathcal{A}$ est un champ aléatoire où la distribution de la variable représentant la valeur du site $s \in S''$ (resp. $s \in S_m$) ne dépend que des valeurs des sites adjacents de $\mathcal{N}_s$ (resp. $\mathcal{O}_s$). (Pour plus de détail on pourra se référer à [FMP03]).

Cela implique que le champ de traces de $\mathcal{A}$ est un *champ de Markov*.

Avant donc de donner formellement cette proposition nous devons donner une définition précise d'un champ de Markov.

Définition 8.5. (Champs de Markov)

Un champ de variables aléatoires $X = (X(s))_{s\in S}$ est un champ de Markov pour le vooisinage N si pour tous les sites $s \in S$, pour tout K n'appartenant pas au voisinage de s (que l'on notera N_s), les variables aléatoires $X(s)$ et $X(k)$ sont indépendantes sachant $X(N_s)$.

Soit en terme mathématique :

$$P(X(s) = x(s)|X(S\backslash s) = x(S\backslash s)) = P(X(s) = x(s)|X(N_s) = x(N_s))$$

Ainsi :

Proposition 8.6. *Pour tout $m > 0$ et tout $T_0 \in \Omega_{S_0}$, le champ aléatoire X de traces de $\mathcal{A}$ de longueur m partant de T_0, est un champ de Markov pour le système de voisinage $\{\mathcal{N}_s\}_{s\in S''} \cup \{\mathcal{O}_s\}_{s\in S^m}$.*
Soit, pour tout $s \in S''$ (resp. $s \in S_m$), tout $q \in Q$ et tout $\eta \in \Omega_{S\setminus\{s\}}$:

$$\mathbb{P}(X(s) = q \mid X(S \setminus \{s\}) = \eta) \quad = \quad \mathbb{P}(X(s) = q \mid X(\mathcal{N}_s) = \eta(\mathcal{N}_s))$$

(resp. $\mathbb{P}(X(s) = q \mid X(S \setminus \{s\}) = \eta) \quad = \quad \mathbb{P}(X(s) = q \mid X(\mathcal{O}_s) = \eta(\mathcal{O}_s))$.

Nous nous attarderons pas sur la preuve simple de se théorème qu'on pourra retrouver dans [FMP03]

8.3 Critère d'auto-stabilisation

8.3.1 Condition de van den Berg

Afin de prouver l'absence de "transition de phase" dans les champs de Markov, van den Berg a introduit un critère basé sur la notion de chemin de désaccord (voir [vdB93]).

L'absence de transition de phase correspond dans notre cadre au cas où $\mathcal{A}$ un une unique distribution stationnaire π (voir [FMP03]). De plus, dans notre contexte, le critère de van den Berg assure non seulement l'unicité de la distribution stationnaire π, mais aussi le mélange rapide de $\mathcal{A}$ vers π, comme nous allons le montrer.

Si $\mathcal{T}$,$\mathcal{U} \in \Omega_S$, on dira que le couple $(\mathcal{T},\mathcal{U})$ a un *chemin de désaccord* de S_0 à S_m si il y a une suite de sites $r_1 \sim \cdots \sim r_k$ dans S avec $r_1 \in S_0$, $r_k \in S_m$ et $\mathcal{T}(r_h) \neq \mathcal{U}(r_h)$ pour $h = 0, \cdots, k$.
Le résultat suivant est vrai dans le cas général des champs de Markov (voir [vdB99] [vdB93]).

Proposition 8.7. *Pour tout $m > 0$, $r \in S_0$, et tout couple $T_0, U_0 \in \Omega_{S_0}$ avec $T_0(s) = U_0(s)$ si $s \neq r$, on a :*

$$d(\mathcal{P}^{\mathcal{A}}_{S',S_m}(.,T_0),\ \mathcal{P}^{\mathcal{A}}_{S',S_m}(.,U_0)) \leq (\mathcal{P}^{\mathcal{A}}_{S'}(.,T_0) \times \mathcal{P}^{\mathcal{A}}_{S'}(.,U_0))$$
$$(\exists \text{ un chemin de désaccord de } S_0 \text{ à } S_m)$$

La proposition 8.7 exprime le fait que la distance en variation (définie au chapitre 6) entre les distributions marginales sur S_m des deux distributions $\mathcal{P}^{\mathcal{A}}_{S'}(.,T_0)$ et $\mathcal{P}^{\mathcal{A}}_{S'}(.,U_0)$ est inférieure à la probabilité que,si on choisit aléatoirement deux éléments de $\Omega_{S'}$, l'un avec la première distribution et l'autre avec la seconde distribution, ce couple d'éléments a un chemin de désaccord de S_0 à S_m.

Van den Berg et Maes ont étendu ce résultat dans le cas où, au lieu de choisir deux éléments indépendamment dans $\Omega_{S'}$, on considère un coupling de ces deux distributions qui vérifie des conditions de monotonie (voir [vdBM94]).

Par contre, ce résultat ne s'étend pas à tous les coupling comme le montre le contre-exemple de Häggström [Hag01].

Montrons juste, dans notre contexte de traces de $\mathcal{A}$, que cette variante du critère de van den Berg s'applique lorsque le coupling (X,Y) est induit d'un coupling localement stable $\{(x_t,y_t)\}_{t\geq 0}$.

De plus, d'un côté le résultat de van den Berg sera étendu car nous nous focaliserons sur les chemins de désaccord S-SE ; et d'un autre côté, il sera affaibli en ne considérant que des états initiaux T_0 et U_0, qui pourrait différer de plus d'un site r.

En utilisant le fait que $\mathcal{P}^{\mathcal{A}}_{S',S_m}(T_m,T_0) = Pr(x_m = T_m \mid x_0 = T_0)$ (voir la proposition 8.3), on adapte donc la proposition 8.7 ainsi :

Proposition 8.8. *Pour tout $m > 0$, tout couple $T_0, U_0 \in \Omega_{S_0}$ et tout coupling localement stable $\{(x_t,y_t)\}_{t\geq 0}$ de $\mathcal{A}$ avec $(x_0,y_0) = (T_0,U_0)$,*

le coupling induit (X,Y) de traces de $\mathcal{A}$ de longueur m (dont les distributions sont notées μ) est tel que :

$$d(Pr(x_m = . \mid x_0 = T_0),\ Pr(y_m = . \mid y_0 = U_0)) \leq \mu(\exists\ S\text{-}SE \text{ chemin de désaccord de } S_0 \text{ à } S_m).$$

La preuve de la proposition 8.8 repose sur le lemme suivant :

Lemme 8.9. *Pour tout $m > 0$, tout couple $T_0, U_0 \in \Omega_{S_0}$ et pour tout coupling localement stable $\{(x_t,y_t)\}_{t\geq 0}$ de $\mathcal{A}$ avec $(x_0,y_0) = (T_0,U_0)$, le coupling induit (X,Y) des traces de $\mathcal{A}$ de longueur m (qui a la distribution notée μ) est tel que :*

$x_m \neq y_m$ ssi il existe un chemin S-SE de désaccord de S_0 à S_m (μ presque sûrement).

La preuve de ce lemme est simple, en effet s'il existe un chemin de désaccord, il est évident que $x_m \neq y_m$, et si $x_m \neq y_m$ le fait que le coupling soit localement stable implique qu on peut remonter cette différence par voisinage jusque la configuration (x_0,y_0) et donc créer un chemin S-SE de désaccord.

Nous pouvons à présent donner la preuve de la proposition 8.8

Démonstration. $d(Pr(x_m = . \mid x^0 = T_0),\ Pr(y_m = . \mid y^0 = U_0))$
$\leq Pr(x_m \neq y_m \mid (x_0, y_0) = (T_0, U_0))$ (par le "lemme de coupling" voir 6.13)
$\leq \mu(\exists$ un chemin de désaccord S-SE de S_0 à $S_m)$ (par le lemme 8.9). □

Définition 8.10. *Un champ de traces de $\mathcal{A}$ est* rapidement accordant *si pour tout $m > 0$ et chaque couple $T_0, U_0 \in \Omega_{S_0}$, il existe un coupling localement stable $\{(x_t, y_t)\}_{t\geq 0}$ de $\mathcal{A}$ avec $(x_0, y_0) = (T_0, U_0)$ tel que le coupling induit (X, Y) de traces de $\mathcal{A}$ de longueur m (dont la distribution est notée μ) vérifie :*

$\mu(\exists$ *un chemin de désaccord S-SE de S_0 à S_m)* $\leq Kexp(-c\ m)$,
où K et $1/c$ sont polynomiaux en N.

On a :

Théorème 8.11. *Si le champ de traces de $\mathcal{A}$ est rapidement accordant, alors $\mathcal{A}$ a une unique distribution stationnaire π, et $\mathcal{A}$ est rapidement mélangeant (pour les définitions on pourra se reporter au chapitre 6).*

Démonstration. Montrons tout d'abord par l'absurde qu'il n'y a qu'une seule distribution stationnaire. S'il y avait deux distributions stationnaires π_1 et π_2, la distance en variation après m étapes entre la distribution issue de la distribution initiale π_1 et la distribution issue de la distribution initiale π_2, est plus grande qu'une constante strictement positive, et ne décroîtra pas exponentiellement avec m (vu que ce sont des distributions stationnaires).

Par la proposition 8.8, cela contredirait l'hypothèse de rapide accordance.

Soit donc π l'unique distribution stationnaire, par la proposition 8.8 et la rapide accordance, on a :

$$d(Pr(x_m = . \mid x_0 = T_0), \pi) \leq \mu(\exists \text{ un chemin de désaccord } S\text{-}SE \text{ de } S_0 \text{ à } S_m) \leq Kexp(-cm).$$

Le temps de mélange $\tau(\varepsilon)$ est ainsi borné par la relation :

$$Kexp(-c\tau(\varepsilon)) \leq \varepsilon,$$

soit $\tau \geq \frac{ln(\varepsilon^{-1})}{c} + \frac{ln(K)}{c}$, qui prouve que τ est polynomial en N et $ln(\varepsilon^{-1})$ et qui achève la preuve.

□

8.4 Applications

8.4.1 Algorithme d'exclusion mutuelle auto-stabilisant de Herman

Revenons une nouvelle fois à l'exemple de l'algorithme de Herman (voir l'exemple 7.6 pour la définition de l'algorithme). Etant donnée une configuration, chaque étape d'exécution de l'algorithme fait chaque jeton soit se déplacer vers la droite avec probabilité $\frac{1}{2}$, ou rester à la même place avec probabilité $\frac{1}{2}$.

Il est très facile de montrer que le nombre de jetons ne peut jamais augmenter. Par contre le nombre de jetons peut diminuer lorsque un jeton entre en collision avec un jeton situé immédiatement à sa droite :
un enchevêtrement de jetons intervient (de la forme 000 ou 111), et à l'étape suivante ces deux jetons disparaissent simultanément avec la probabilité 1/4.

L'évolution de la distance entre deux jetons consécutifs, peut alors être modélisée comme une marche aléatoire sur un intervalle ; ainsi l'éventuelle collision de deux jetons est un événement de probabilité 1, et tous les jetons sauf 1 vont disparaître presque sûrement.
L'ensemble $\mathcal{L}$ des configurations avec un seul jeton correspond à l'ensemble des "configurations légales" (voir [Dij74]), et est ergodique.
Il est aisé de montrer que la distribution stationnaire π associée à ce processus est la distribution uniforme sur les éléments de $\mathcal{L}$. (Plus précisément, $\pi(\overline{q}) = 1/(2N)$ si $\overline{q} \in \mathcal{L}$, et $\pi(\overline{q}) = 0$ sinon.)

Un tel ensemble $\mathcal{L}$ ergodique est unique (et par la même, une telle distribution π aussi), ce qui correspond bien à la notion d'auto-stabilisation définie dans le chapitre précédent
Nous allons montrer à présent :

Théorème 8.12. *Le champ de traces de l'algorithme de Herman est rapidement accordant.*

Par le théorème 8.11, cela nous donnera une nouvelle preuve de l'auto-stabilisation pour Herman.
De plus, le théorème 8.11 garantit que la distribution sur les configurations est "ε-proche" de la distribution uniforme π sur les distributions légales après N^2 étapes. Nous donnons donc ici une nouvelle valeur de temps de convergence celle de temps de mélange que nous n'avions pas borné auparavant.

Démonstration. L'idée de la preuve du théorème 8.12 est la suivante.
Considérons de traces de $\mathcal{A}$ X et Y induite par le coupling (x, y) de l'exemple 7.6, qui ne sont pas en accord sur des sites de S_0 ($x_0 \neq y_0$).
Calculons la probabilité qu'il existe un chemin de désaccord sur (X, Y).

Soit deux positions extrêmes de désaccord i_0 et j_0 de S_0, i.e. des sites de désaccord ($x_0(i_0) \neq y_0(i_0)$, $x_0(j_0) \neq y_0(j_0)$) tous les sites de S_0 en dehors du segment $[i_0, j_0]$ ou à l'intérieur de $]i_0, j_0[$ soit en accord. Supposons, sans perte de généralité que : $1 \leq i_0 \leq j_0 \leq N$ et $x^0(k) = y^0(k)$, pour tout $k, 1 \leq k \leq i_0 - 1$ ou $j_0 + 1 \leq k \leq N$.

Par stabilité locale, tous les sites entre $j_0 + 2$ et $i_0 - 1$ restent en accord après l'étape 1 : pour tout k avec $0 \leq k \leq i_0 - 1$ ou $j_0 + 2 \leq k \leq N$, $(x, y)_1(k) = (q_k, q_k)$ avec $q_k \in Q$.
Considérons à présent les positions limites i_0 et $j_0 + 1$.

En i_0, la valeur de $(x, y)_1$ dépend des valeurs de $(x, y)_0$ au sites $i_0 - 1$ et i_0. Ainsi : $(x, y)_0(i_0 - 1) = (q, q)$ et $(x, y)_0(i_0) = (q', \neg q')$ avec $q, q' \in Q$. Afin de simplifier et de se fixer les idées, supposons : $(x, y)^0(i_0 - 1) = (0, 0)$ et $(x, y)^0(i_0) = (0, 1)$. (Les autres cas sont

similaires.)

A la position i_0, il y a en jeton en x_0 (rappelons que ça implique $x_0(i-1) = x_0(i)$ $(= 0$ dans ce cas)) mais pas de jetons en y_0. La valeur de $x_1(i_0)$ est 0 (resp. 1) avec probabilité $\frac{1}{2}$ alors que la valeur de $y_1(i_0)$ est 0 avec probabilité 1. Ainsi x_1 et y_1 sont en accord (resp. désaccord) en i_0 avec probabilité $\frac{1}{2}$.
En cas d'accord, la position du site extrême de désaccord de gauche se déplace d'(au moins) un site vers la droite, et devient $i_0 + 1$ en S_1 (un mouvement *SE*).
En cas de désaccord, il reste égal à i_0 dans S_1 (un mouvement S).

De la même manière, au site d'accord situé le plus à gauche $j_0 + 1$, il y a un jeton en x_0 et aucun jeton en y_0, ou l'inverse. Ainsi, à l'étape 1, x_1 et y_1 sont en désaccord (resp. accord) au site $j_0 + 1$ avec probabilité $\frac{1}{2}$.
En cas de désaccord, le site extrême de désaccord de droite se déplace d'un site vers la droite, et devient $j_0 + 1$ dans S_1 (mouvement *SE*).
En cas d'accord, il reste au plus égal à j_0 (mouvement S).

De ce fait le site de désaccord le plus à gauche suit un mouvement S avec probabilité $\frac{1}{2}$, ou (dans le pire des cas) un mouvement *SE* avec probabilité $\frac{1}{2}$ de S_0 à S_1.
Identiquement le site de désaccord le plus à gauche suit les mêmes mouvements.

Itérons à présent ce raisonnement, nous définissons de chemins *S-SE* extrêmes de désaccord $L : l_0 \sim l_1, \cdots$ et $R : r_0 \sim r_1, \cdots$ jusqu'à ce que l'un des cas limites se produise : cas de désaccord total ou cas d'accord total.

Le cas de désaccord total apparaît lorsque, à l'étape t, aucun site n'est en accord entre x et y ($l_t = (t, i_t), r_t = (t, j_t)$ avec $j_t = i_t + (N-1)$).
Comme N est impair, il y a toujours au moins un jeton dans x_t. En cas de désaccord total, la position du jeton, disons 00, dans x_t correspond correspond à une position d'un jeton 11 dans y_t.
Alors, à l'étape $t+1$, ses positions seront en accord avec probabilité 1, grâce à notre coupling. Le cas de désaccord total après t étapes est ainsi transitoire, car il y a toujours un site en accord à l'étape $t + 1$.

L'accord total après t étapes (tous les sites sont en accord dans $(x, y)_t$), est alors définitif. En effet cela correspond à l'arrêt définitif de L et R (plus aucun chemin de désaccord n'apparaîtra) : tous les sites seront en accord entre t et m.

Le chemin *S-SE* de désaccord de gauche (resp. de droite) L (resp. R) est représenté en gras (resp. pointillé) dans la figure 8.5. Tous les sites sont en accord en dehors de la zone que ces deux chemins délimitent. A l'intérieur de la zone, les chemins *S-SE* sont généralement (mais non nécessairement) des chemins de désaccord. Cette zone est appelée *zone de désaccord.*

Chaque chemin de désaccord dans (X, Y) est un chemin *S-SE*de longueur m et de probabilité $(1/2)^m$.
Afin d'évaluer (une borne supérieure de) la probabilité Pr qu'il existe un chemin de désaccord, il suffit d'évaluer la probabilité d'existence d'un couple (L, R) de chemins *S-SE* de désaccord

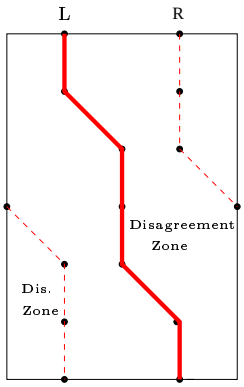


FIG. 8.5 – Zone de désaccord

de longueur m. Cela, comme nous allons le voir, se réduit à évaluer le nombre de tels chemins (L, R).

Sans perte de généralité on suppose que L commence à la position $i_0 = 1$ de S_0.
Nous allons transformer L en un chemin L' (partant de $i_0 = 1$) fait d'étapes S uniquement ; de ce fait, il faudra transformer R en un chemin R' à la même distance de L' que R de L. Le chemin R' ne sera donc plus uniquement composé de mouvements S et SE, mais aussi de mouvements SW (i.e., mouvements d'un site (t, j) à un site $(t + 1, j - 1)$).
Nous procédons donc à la transformation géométrique suivante :

a. Si l'étape t de L et R est un mouvement S pour les deux, alors l'étape t pour L' et R' sont des mouvements S :

b. Si l'étape t de L et R est un mouvement SE pour les deux, alors l'étape t pour L' et R' sont des mouvements S :

c. Si l'étape t de L (resp. R) est un mouvement S (resp. un mouvement SE), alors l'étape t de L' (resp. R') est un mouvement S (resp. un mouvement SE) :

d. Si l'étape t de L (resp. R) est un mouvement SE (resp. un mouvement S), alors l'étape t de L' (resp. R') est un mouvement S (resp. un mouvement SW) :

Après cette transformation, chaque chemin *S-SE* R de la zone de désaccord initiale devient un chemin *S-SE-SW* R' dans le rectangle de longueur m et largeur N.

Dans la figure originale la probabilité associée avec chaque étape de couple de chemins (L, R) était toujours $\frac{1}{2} \times \frac{1}{2} = \frac{1}{4}$.
Dans la figure transformée la probabilité associée à une étape (L', R') est $\frac{1}{4}$ lorsque cela correspond à un mouvement *SE* ou *SW* de R', et $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$ lorsque cela correspond à une étape S de R' (à cause de la redondance des cas a et b).
Par exemple, les chemins L et R de la figure 8.5 se transforment en L' et R' dans la figure 8.6.

En cas de désaccord total à l'étape t, la transformation exacte de (L, R) en (L', R') devrait nous donner un mouvement *SW* de probabilité 1 pour R' (voir la section 8.4.1).
Pour l'homogénéité, nous donnerons à ce mouvement *SW* une probabilité 1/4 et au mouvement S une probabilité 1/2, ce qui contribue à une légère sur-approximation de l'événement d'existence d'un chemin de désaccord.

L'évaluation du nombre de chemins de désaccord se réduit donc au problème du comptage de tous les chemins *S-SE-SW* inclus dans le rectangle de longueur m et largeur N.

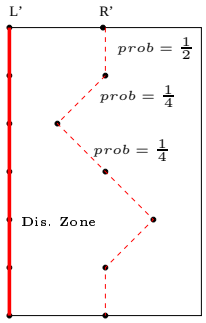


FIG. 8.6 – Zone de désaccord après la transformation

Tout d'abord nous ferons abstraction des mouvements S de R', et nous calculerons le nombre de chemins *SE-SW* de longueur n (avec $n \leq m$) (voir figure 8.7).

Lemme 8.13. *Soit $U_{i,n}$ le nombre de chemins* SE-SW *de longueur n partant du site $(0, i)$ dans une zone rectangulaire de largeur N. Il existe une constant A_N telle que, pour tout $i \in [1, \cdots, N]$ et tout $n > 0$:*

$$U_{i,n} \leq A_N (2 \cos \frac{\pi}{N+1})^n.$$

Démonstration. Soit $U_n = (U_{1,n}, \cdots, U_{i,n}, \cdots, U_{N,n})$. La suite $(U_n)_{n \geq 0}$ vérifie l'équation de récurrence linéaire : $U_{n+1} = PU_n$ où la matrice $N \times N$ P est définie par :

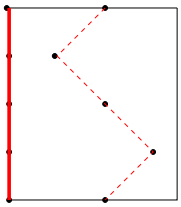


FIG. 8.7 – Zone rectangulaire après le retrait des mouvements S

$$P = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 1 & 0 & 1 & \dots & 0 & 0 \\ 0 & 1 & 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & 0 & 1 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}$$

Les valeurs propres de cette matrice sont $(2\cos\frac{k\pi}{N+1})_{1\leq k\leq N}$. De plus, pour tout i dans $[1,N]$, il existe des constantes $A_{i,1},\cdots,A_{i,N}$ telles que, pour tout $n>0$:
$U_{i,n} = \sum_{k=1}^{N} A_{i,k}(2\cos\frac{k\pi}{N+1})^n$.
On pose : $A_N = max\{\sum_{k=1}^{N}|A_{i,k}| \ ; \ i\in[1,N]\}$. □

Proposition 8.14. *Notons* $\mu_N = \frac{1+\cos\frac{\pi}{N+1}}{2} = (\cos\frac{\pi}{2(N+1)})^2$. *Il existe une constante* K_N *telle que, pour tout* $m>0$ *:*

$$Pr(\exists \text{ un chemin de désaccord de longueur } \ m) \leq K_N(\mu_N)^m.$$

Démonstration. Soit $\lambda_N = 2\cos\frac{\pi}{N+1}$. Soit $n\leq m$. Dans la zone rectangulaire $[1,\cdots,m]\times[1,\cdots,N]$, le nombre de chemins *S-SE-SW* de longueur m, partant de $(0,i)$, avec exactement $m-n$ mouvements S est $C_m^{m-n}U_{i,n}$. (nous choisissons les $m-n$ mouvements S et les retirons afin d'obtenir un chemin *SE-SW* de longueur n.)

La probabilité d'un tel chemin est $\frac{1}{4}^n\frac{1}{2}^{m-n}$. De ce fait, en appliquant le lemme 8.13, nous avons :

$$\begin{aligned} Pr(\exists \text{ un chemin de désaccord de longueur } \ m) &\leq A_N\sum_{n=0}^{m} C_m^{m-n}\frac{1}{4}^n\frac{1}{2}^{m-n}\lambda_N^n \\ &= A_N(\frac{\lambda_N}{4}+\frac{1}{2})^m \\ &= A_N(\frac{\lambda_N+2}{4})^m \\ &= A_N(\mu_N)^m. \end{aligned}$$

□

On peut, à présent, calculer le temps de mélange.

Lemme 8.15. *Lorsque N tend vers ∞, on a :* $-ln\mu_N \sim \frac{\pi^2}{4N^2}$.

Proposition 8.16. *L'algorithme de Herman est rapidement accordant.*
De plus le temps de mélange est quadratique en N.

Démonstration. $Pr(\exists$ un chemin de désaccord de longueur $t) \leq K_N exp(-ct)$,
où $c = -ln(\mu_N) \sim \frac{\pi^2}{4N^2}$ (par le lemme 8.15). Ainsi par la définition 8.10, l'algorithme de Herman est rapidement accordant.
Par le théorème 8.11, le temps de mélange est quadratique (comme $1/c$).

□

□

8.4.2 Marche aléatoire sur un anneau.

L'exemple précédent correspond à un algorithme issu de la communauté des algorithmes distribués. Une propriété intéressante de cet algorithme est que la chaîne de Markov correspondante est non-reversible.
Nous allons montrer à présent que notre méthode combinatoire du calcul des chemins de désaccord s'applique tout aussi bien pour évaluer le temps de mélange des chaînes de Markov plus classiques (réversible). Nous l'illustrons par la l'exemple simple de la marche aléatoire sur un anneau. Le temps de mélange est quadratique (voir Diaconis et Saloff-Coste [DSC96], p. 732).

Chaque configuration est de la forme $(0, \cdots, 0, 1, 0, \cdots, 0)$. La configuration est déterminée par la position d'un unique 1.
La transition probabiliste est de la forme :

`SI` $(x_{i-1}, x_i, x_{i+1}) = (0,1,0)$ `ALORS` $(x'_{i-1}, x'_i, x'_{i+1}) = (1,0,0)$ (resp. $(0,0,1)$)
avec probabilité $\frac{1}{2}$.

Même si cette forme de transition est différente de celle décrite dans la section 7.1.6 (car la mise à jour dépend aussi de la valeur courante du voisin de droite), il est aisè d'étendre notre méthode de chemins de désaccord d'une manière directe. (En particulier, la proposition 8.8 reste vraie, sous la condition de remplacer les chemins *S-SE*par la notion de chemin *S-SE-SW*.)

Expliquons à présent comment la méthode de chemins de désaccord nous permet de retrouver la borne quadratique.
Nous définissons un coupling localement stable comme suit. Pour tout $1 \leq i \leq N$, pour tout $t \geq 0$:

Si $(x,y)_t(i) = (1,1)$ alors $(x,y)_{t+1}(i+1) = (1,1)$ avec probabilité $\frac{1}{2}$, ou
$(x,y)_{t+1}(i-1) = (1,1)$ avec probabilité $\frac{1}{2}$.

On a :

Théorème 8.17. *Le champ de traces de la marche aléatoire sur un anneau est rapidement accordant*

L'idée de la preuve est similaire à celle de l'algorithme de Herman : on estime le nombre de chemins de désaccord (de longueur m) sur le coupling induit (X, Y) de traces de $\mathcal{A}$-en introduisant la même notion de zone de désaccord.

Les chemins de désaccord sont ici des chemins *S-SE-SW*. Le problème est donc d'énumérer le nombre de chemins *S-SE-SW*.Montrons de la même manière que le calcul nous permet de trouver une borne quadratique.

Démonstration. Enumérons le nombre de chemins de désaccord extrêmes *S-SE-SW*.

Comme précedemment, nous définissons deux chemins *S-SE-SW* de désaccord de longueur m, $L : l_1 \sim l_2 \sim \cdots \sim l_m$ et $R : r_1 \sim r_2 \sim \cdots \sim r_m$.
Similairement, on transforme chaque couple de chemins (L, R) en un couple (L', R') tel que L' est uniquement composé de mouvements S, et R' est composé de mouvements de la forme : S, SEE, SWW. (Un mouvement SEE se rend d'un site (t, i) à un site $(t+1, i+2)$, et similairement pour SWW.)

On considère à présent l'ensemble des chemins *S-SEE-SWW* dans la zone rectangulaire de longueur m et de largeur N. Notons que pour ces chemins la probabilité d'un mouvement SEE SWW est $\frac{1}{4}$ mais la probabilité d'un mouvement S est $\frac{1}{4} + \frac{1}{4} = \frac{1}{2}$.

Soit $V_{i,n}$ le nombre de chemins *SEE-SWW* de longueur n partant du site $(0, i)$ dans une zone rectangulaire de largeur N.
Nous avons la relation de récurrence suivante :

$V_{i,n} = V_{i-2,n-1} + V_{i+2,n-1} \quad \forall\ 2 \leq i \leq N-2,$
Ainsi les deux suites :
$V_n^{\text{odd}} = (V_{1,n}, \cdots, V_{2i+1,n}, \cdots) = (V_{2i+1,n})_{1 \leq 2i+1 \leq N}$ et
$V_n^{\text{even}} = (V_{2,n}, \cdots, V_{2i,n}, \cdots) = (V_{2i,n})_{1 \leq 2i \leq N}$ vérifient indépendamment la même relation de récurrence que la suite U_n du lemme 8.13.
Enfin, en notant M la partie entière de $(N+1)/2$, on obtient les résultats suivants :

Lemme 8.18. *Il existe une constante B_N telle que, pour tout $i \in [1, \cdots, N]$ et tout $n > 0$:*

$$V_{i,n} \leq B_N (2 \cos \frac{\pi}{M+1})^n.$$

Proposition 8.19. *Notons $\nu_N = \frac{1+\cos\frac{\pi}{M+1}}{2} = (\cos\frac{\pi}{2(M+1)})^2$. Il existe une constante I_N telle que, pour tout $m > 0$:*

$$Pr(\exists \text{ un chemin de désaccord de longueur } \quad m) \leq I_N (\nu_N)^m.$$

On peut à présent calculer le temps de mélange.

Lemme 8.20. *Lorsque N tend vers ∞, on a :* $-ln\nu_N \sim \frac{\pi^2}{N^2}$.

Proposition 8.21. *La marche aléatoire classique sur un anneau est rapidement accordante. De plus son temps de mélange est quadratique.*

Démonstration. $Pr(\exists$ un chemin de désaccord de longueur $t) \leq I_N exp(-ct)$,
où $c = -ln(\nu_N) \sim \frac{\pi^2}{N^2}$. Ainsi, par la définition 8.10, La marche aléatoire sur un anneau est rapidement accordante. Par le théorème 8.11, le temps de mélange est quadratique (comme $1/c$). □

□

Quatrième partie

Perspectives

Chapitre 9

Extensions et perspectives

La bataille contre l'ignorance se gagne tous les jours, et elle finit par ouvrir sur des perspectives insoupçonnées.
Tenzin Gyatso (Le Dalaï-Lama)

9.1 Extensions du coupling

Comme nous l'avons vu dans le chapitre précédent, nos critères d'applications de méthodes liés au coupling souffrent de certaines restrictions. En effet, elles ne s'appliquent que dans le cas où $\mathcal{L}$ est fortement connexe. De plus, il semble être intéressant de considérer des extensions de ces techniques au cas où la politique n'est pas fixée (cas des processus de décision markoviens). L'extension au cas où $\mathcal{L}$ n'est pas connexe est facilement surmontable, en utilisant un niveau supplémentaire d'abstraction (cf [FMP04a]). Les autres extensions sont, elles, des perspectives que nous discutons.

9.1.1 Cas où $\mathcal{L}$ n'est pas connexe

Jusqu'à présent, nous avons appliqué nos méthodes dans le cas où l'algorithme $\mathcal{A}$ (vu comme une chaîne de Markov) a un unique ensemble ergodique.
On peut donner une extension du théorème 7.16 qui s'applique lorsque $\mathcal{A}$ a plus d'un ensemble ergodique. (Le théorème 7.14 s'étend de manière complètement similaire.)

Théorème 9.1. *Soit une chaîne de Markov $\mathcal{A}$ et deux ensembles ergodiques clos et disjoints $\mathcal{L}_0$ et $\mathcal{L}_1$ ($\mathcal{L} = \mathcal{L}_0 \uplus \mathcal{L}_1$), supposons qu'il existe un coupling (X_t, Y_t) et une fonction δ sur $\Omega \times \Omega$ à valeurs dans $\{0, 1, \cdots, B\}$ tels que :*

- $\delta(X_t, Y_t) = 0$ *ssi* $X_t = Y_t$,
- $\delta(X_t, Y_t) = B \Rightarrow \delta(X_{t+1}, Y_{t+1}) = B$,
- $(\delta(X_t, Y_t) = B \wedge Y_t \in \mathcal{L}_0) \Rightarrow X_t \in \mathcal{L}_1$,
- *il existe* $\alpha > 0$ *tel que, pour tout* (X_t, Y_t) *avec* $0 < \delta(X_t, Y_t) < B$:
  $E[\delta(X_{t+1}, Y_{t+1})] \leq \delta(X_t, Y_t)$
  $\wedge \quad Pr(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq \alpha.$

Alors

1. *$\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}$. De plus :*
2. *Le temps moyen de hitting satisfait :* $\mathbf{H}_{\mathcal{L}} \leq B^2/\alpha$.
3. *Le temps d'ε-absorption satisfait :*
$$\mathbf{\Theta}_{\mathcal{L}}(\varepsilon) \leq \lceil e\tfrac{B^2}{\alpha}\rceil\lceil \ln(\tfrac{1}{\varepsilon})\rceil.$$

La preuve du théorème 9.1 est analogue à celle du théorème 7.16, mais repose sur la proposition suivante :

Proposition 9.2. *Supposons que $D = (D_t)_{t=0}^{\infty}$ est un processus stochastique positif sur $\{0, 1, \cdots, B\}$ tel que $E[D_{t+1}] \leq D_t$.*

De plus supposons que $Pr(D_{t+1} \neq D_t) \geq \alpha$ (avec $\alpha > 0$) lorsque $0 < D_t < B$. Alors si τ est le premier temps tel que $D_t = 0 \vee D_t = B$, on a : $E[\tau] \leq B^2/\alpha$.

Démonstration. Soit D'_t définie par : $D'_t = \begin{cases} D_t & \text{if } D_t \neq B, \\ 0 & \text{if } D_t = B. \end{cases}$

Soit τ' le premier temps auquel D'_t atteint 0. On a $\tau' = \tau$ car $D'_t = 0$ ssi $D_t = 0 \vee D_t = B$. D'un autre côté il est évident de remarquer que D'_t vérifie les hypothèses de la proposition 7.17.
Ainsi, on a $E[\tau] = E[\tau'] \leq B^2/\alpha$.

□

Preuve du Théorème 9.1.

Considérons une fonction à valeurs entières δ vérifiant les hypothèses du théorème 9.1, et montrons les conclusions 2 et 3. (Le fait que $\mathcal{A}$ est auto-stabilisant est une conséquence de la conclusion 2, grâce à la proposition 7.10.)

1. Considérons deux éléments x, y de Ω et un coupling (X_t, Y_t) d'élément initial $(X_0, Y_0) = (x, y)$ tels que $y \in \mathcal{L}_0$.
Soit $D_t = \delta(X_t, Y_t)$ pour $t \geq 0$. Comme $D_t = 0$ ssi $X_t = Y_t$ et $(D_t = B \wedge Y_t \in \mathcal{L}_0) \Rightarrow X_t \in \mathcal{L}_1$, le temps moyen $E[\tau]$ nécessaire à D_t pour atteindre 0 ou B, est une borne supérieure du temps moyen de hitting $\mathbf{H}_{\mathcal{L}}$.
Ainsi, par proposition 9.2, on a :
pour tout $x, y \in \Omega$, $E[\tau] \leq B^2/\alpha$, on a : $\mathbf{H}_{\mathcal{L}} \leq B^2/\alpha$.

2. Soit $D_t = \delta(X_t, Y_t)$. On peut aisément vérifier que D_t vérifie les hypothèses de la proposition 9.2. Soit $T'_{x,y}$ le premier temps tel que $D_t = 0 \vee D_t = B$, partant de $D_0 = \delta(x, y)$ (i.e., $X_0 = x, Y_0 = y$). La proposition 9.2 nous donne :
$E[T'_{x,y}] \leq B^2/\alpha$.
Soit $T = \lceil eB^2/\alpha \rceil$, alors par l'inégalité de Markov, on a la probabilité que $T'_{x,y} > T$ est au plus e^{-1}. Si on lance s exécutions indépendantes de longueur T alors la probabilité de l'événement $D_t \neq 0 \wedge D_t \neq B$ pour $t \geq sT$ est au plus e^{-s}.
Ainsi, pour $t > T\lceil \ln(\varepsilon^{-1})\rceil$, la probabilité de l'événement $D_t = 0 \vee D_t = B$ est au moins $1 - \varepsilon$.
Supposons que $Y_0 \in \mathcal{L}_0$. Alors $Y_t \in \mathcal{L}_0$ (car $\mathcal{L}_0$ est clos). Ainsi $D_t = 0$ implique $X_t \in \mathcal{L}_0$, alors que $D_t = B$ implique $X_t \in \mathcal{L}_1$ (car δ vérifie les hypothèses du théorème 9.1).

Ainsi $X_t \in \mathcal{L}$ dès que $D_t = 0 \vee D_t = B$.
De ce fait, pour tout $X_0 \in \Omega$ et tout $t \geq T\lceil \ln(\varepsilon^{-1}) \rceil$:
$Pr(X_t \in \mathcal{L}) \geq 1 - \varepsilon$. □

Comme dans la section 7.5, il suffit, grâce au lemme 7.19,de vérifier l'hypothèse : $E[\delta(X_{t+1}, Y_{t+1})] \leq \delta(X_t, Y_t)$ du théorème 9.1 sur les couples de configurations voisines uniquement.

Nous pouvons donc appliquer cette extension sur un exemple avec deux ensembles ergodiques disjoints. De plus, il se trouve que cet exemple est décrit sur une autre topologie (graphe). Cet algorithme est un algorithme de consensus qui a pour but d'accorder sur une valeur l'ensemble des machines situées au sommet d'un graphe. En voici la description

Exemple 9.3. *Considérons un graphe régulier non-orienté $G = (V, E)$, avec N sommets et de degré Δ, et une palette de couleurs $C = \{0, \cdots, q-1\}$.On suppose dans cet exemple que q vaut 2, mais on pourrait aisément étendre le raisonnement. Un 0-coloriage (resp. 1-coloriage) est une fonction de V dans $\{0\}$ (resp. $\{1\}$).*
Soit $\mathcal{L}_0$ (resp. $\mathcal{L}_1$) le 0-coloriage (resp. 1-coloriage) de G.
Partant d'un coloriage quelconque de G, considérons l'algorithme $\mathcal{A}$ suivant (avec une politique centrale aléatoire) :

- *On choisit un sommet $v \in V$ et un voisin w de v de manière aléatoire uniforme .*
- *On recolore v avec la couleur de w.*

Bien entendu, le coupling que nous prendrons est le coupling naturel consistant en le même choix de v et de w pour les deux processus.

Soit $\delta(X_t, Y_t)$ le nombre de sommets sur lesquels X_t, Y_t différent, Il est facile de montrer que pour des coloriages adjacents ($\delta = 1$) (X_t, Y_t) :
$E[\delta(X_{t+1}, Y_{t+1})] = \delta(X_t, Y_t)$. (♣)
En effet, si on choisit le site qui dont les couleurs différent pour v alors δ passer à 0 et si on le choisit pour w alors δ passera à 2 sinon il restera inchangé.

De plus, pour chaque couple de coloriages (X_t, Y_t) tels que $0 < \delta(X_t, Y_t) < N$ on a :
$Pr(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq \frac{1}{\Delta N}$ (♣′)

On peut induire de (♣) et (♣′), en utilisant le théorème 9.1, que, partant d'une configuration arbitraire Ω, $\mathcal{A}$ est auto-stabilisant vers $\mathcal{L}_0 \uplus \mathcal{L}_1$, et le temps moyen de hitting est cubique : $\mathbf{H}_{\mathcal{L}} \leq \Delta N^3$. ♣

Le théorème 9.1 peut ainsi être utilisé pour traiter le cas où $\mathcal{L}$ est composé de plus de deux ensembles ergodiques, en rassemblant entre eux des configurations, cette technique est similaire à celle du "lumping" développé dans ([DFP01a]).

A présent, supposons que dans l'exemple 9.3, il y ait $q > 2$ couleurs. Dans ce cas $\mathcal{L}$ est composé de q ensembles ergodiques, qui correspondent aux coloriages monochromatiques.
On pourrait raisonner de la même manière que dans l'exemple 9.3, mais avec $\mathcal{L}_0$ et $\mathcal{L}'_1$ (au lieu de $\mathcal{L}_0$ et $\mathcal{L}_1$), où $\mathcal{L}'_1$ est l'ensemble de tous les coloriages sans la couleur 0.
L'application du théorème 9.1, nous donne que le temps moyen de hitting est encore cubique : Partant d'une configuration arbitraire, on atteint $\mathcal{L}_0$ ou $\mathcal{L}'_1$, en au plus ΔN^3 étapes en moyenne.

Ainsi, chaque coloriage perd au moins une couleur en au plus ΔN^3 étapes en moyenne. Le temps moyen de hitting pour atteindre une configuration monochromatique est donc au plus $(q-1)\Delta N^3$.

9.2 Perspectives

Dans cette dernière partie, nous donnerons une liste non exhaustive des domaines sur lesquels on pourrait étendre nos résultats. Nous donnerons aussi certaines pistes que nous explorons actuellement ou envisageons pour le futur.

9.2.1 Autres topologies

Les exemples que nous avons présentés sont pour la plupart pris sur des topologies en anneau. En fait, comme illustré dans la partie 9.1.1, notre méthode s'applique à des topologies quelconques. Nous envisageons de généraliser son utilisation à d'autres exemples avec des topologies de graphes (voir exemple 9.3).
Les méthodes de coupling s'appliquent aussi à des algorithmes dont la communication se fait par message entre voisins et non pas par lecture d'états. Un exemple de son utilisation dans ce cadre est donné dans [DHT04]

9.2.2 Coupling et processus de décision markoviens

Les processus de décision markoviens modélisent mieux que les chaînes de Markov un très grand nombre d'algorithmes distribués (ceux dont la politique n'est pas déterminée). Il est donc important d'étendre nos méthodes dans ce cadre.
Une méthode naïve qui consisterait à ajouter des quantificateurs universels sur tous les choix possibles ne peut convenir.
Cette extension du coupling est donc un objectif ambitieux et prometteur quant au nombre d'exemples qui pourraient être traités grâce à elle.

9.2.3 Recherche de politiques optimales

Pour les problèmes d'accessibilité (plus généralement, dans le cas des formules exprimées dans la logique PCTL), il est montré dans [BdA95] qu'il existe une politique optimale déterministe et markovienne qui permet d'obtenir la valeur de la probabilité maximale (ou minimale). Ce résultat permettrait de pallier aux problèmes liés à l'extension du coupling au processus de décision markoviens (section 9.2.2). En effet on pourrait imaginer un algorithme de la forme :

1. Trouver une politique optimale pour une propriété donnée

2. Etudier la chaîne de Markov correspondant à cette politique.

Il existe des algorithmes numériques qui permettent de calculer de manière empirique la politique optimale dans le cas du problème du plus court chemin stochastique [BT91]. Cet algorithme empirique a été implémenté dans un stage au LSV [Hug04], et a permis de faire des conjectures sur les politiques optimales dans un grand nombre d'exemples, qu'on peut alors

démontrer effectivement.

Il serait par exemple intéressant d'essayer de deviner effectivement les politiques extrémales pour notre modélisation de CSMA/CD afin de vérifier le modèle efficacement grâce à APMC.

Chapitre 10

Conclusion

Le secret d'un bon discours, c'est d'avoir une bonne introduction et une bonne conclusion. Ensuite, il faut s'arranger pour que ces deux parties ne soient pas très éloignées l'une de l'autre.
George Burns

Cette thèse nous a permis de présenter une vue étendue des méthodes probabilistes pour la vérification des systèmes distribués. Nous nous sommes penchés particulièrement sur les problèmes d'accessibilité pour les processus de décision markoviens et les problèmes de convergence pour les chaînes de Markov.

Voici les résultats principaux obtenus :

1. **Etude de cas réel : CSMA/CD par analyse à base d'automates temporisés probabilistes :**
   Nous avons modélisé ce protocole sous forme d'automates temporisés probabilistes. Nous avons étudié de manière quantitative deux modèles différents.
   Le premier, classique (de Nicollin, Sifakis et Yovine), nous a permis d'utiliser les techniques de model-checking des automates temporisés probabilistes en combinant les utilisations de HyTech et PRISM.
   Le second, plus élaboré, a été étudié grâce aux model-checkers PRISM et APMC, dont les avantages respectifs ont été comparés pour la première fois.
   Nous avons ainsi vérifié un certain nombre de propriétés d'accessibilité comme : le nombre moyen de collisions, la probabilité qu'un message soit correctement envoyé avant un certain délai ou le temps moyen avant la livraison correcte d'un message.

2. **Utilisation originale de la technique de coupling pour la preuve de convergence :**
   Nous avons élaboré une nouvelle technique de preuve de convergence pour les algorithmes distribués probabilistes. Dans ce cadre, nous avons exhibé un critère de convergence pour les algorithmes auto-stabilisants. Ce critère et son raffinement à l'aide du path-coupling est simple et performant. Il donne de surcroît une majoration du temps de convergence.

3. **Application de la technique de path-coupling au problème du prisonnier itéré**
   De meilleures bornes pour le temps de convergence de cet algorithme ont été obtenues grâce au path-coupling.

4. **Application du coupling à un problème de Physique statistique : l'ASEP**
   Nous avons donné une preuve élégante de la convergence vers la stationnarité, et obtenu une borne quadratique du temps de convergence pour un cas particulier d'un problème classique de flux de particules (asymmetric simple exclusion problem) grâce à un coupling.

5. **Application de la technique du chemin de désaccord à la preuve de convergence :**
   Nous avons poussé les analogies entre l'algorithmique distribuée et la Physique statistique. Le fait de considérer les algorithmes distribués probabilistes comme des champs de Markov nous a permis de donner des critères de convergence d'algorithmes autostabilisants probabilistes basés sur la notion de chemin de désaccord.

Cette thèse, nous l'espérons, donne un panorama varié du problème de la modélisation et vérification des systèmes probabilistes en exploitant, d'une part, des méthodes qui arrivent à maturation comme le model-checking probabiliste et l'outil PRISM, et en jetant, d'autre part, des passerelles avec la Physique statistique.

Bibliographie

[ACD91] R. Alur, C. Courcoubetis, and D.L. Dill. Model checking for probabilistic real-time systems. In *Proc. of the 18th Int. Conf. on Automata, Languages and Programming (ICALP91)*, pages 115–136, 1991.

[AD94] R. Alur and D. Dill. A theory of timed-automata. *Theoretical computer science*, 126 :183–235, 1994.

[AFar] A. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graph.* draft at http :/www.stat.Berkeley.EDU/users/aldous/book.html, To appear.

[AMT98] E. Asarin, O. Maler, and S. Tripakis. On discretization of delays in timed automata and digital circuits. In *Sangiorgi and de Simone (SdS98)*, pages 470–484, 1998.

[APM] APMC. APMC web page. http ://www.lri.fr/ syp/APMC.

[Ave] Averroes. Averroes web page. http ://www-verimag.imag.fr/AVERROES/.

[Axe84] R. Axelrod. *The evolution of cooperation.* Basic Book, New York, 1984.

[Bai98] C. Baier. *On algorithmic verification for probabilistic systems.* PhD thesis, 1998.

[BD97] R. Bubley and M. Dyer. Path coupling : A technique for proving rapid mixing in Markov chains. In *Proc. of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS'97)*, pages 223–231, 1997.

[BdA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. 15th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.

[BDLGJ02] J. Beauquier, J. Durand-Lose, M. Gradinariu, and C. Johnen. Token based self-stabilizing uniform algorithms. *Journal of Parallel and Distributed Computing*, 62(5) :899–921, 2002.

[Ber95] D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I and II. Athena Scientific, 1995.

[Ber99] M. Bernardo. *Theory and application of extended markovian process algebra.* PhD thesis, University of Bologna, 1999.

[Bey01] D. Beyer. Improvements in bdd-based reachability analysis of timed automata. In *Proc. 10th International Symposium on Formal Methods Europe (FME2001)*, pages 318–343. LNCS 2021, 2001.

[BK96] C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. Technical report, University of Birmingham, June 1996.

[BMK88] D.R. Bogs, J.C. Mogul, and C.A. Kent. Measure capacity of an ethernet : myths and reality. In *Proc. of the SIGCOMM'98 Symposium on COmmunications Architectures and Protocols*, 1988.

[BMT99] M. Bozga, O. Maler, and S. Tripakis. Efficient verification of timed automata using dense and discrete time semantics. In *Proc. 11th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME99)*, pages 125–141. LNCS 1703, 1999.

[Bré99] P. Brémaud. *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues.* Springer, 1999.

[BT91] D.P. Bertsekas and J.N. Tsitsiklis. An analysis of stochastic shortest path problems. *Math. of Op. Res.*, 16(3) :580–595, 1991.

[CDMM02] S. Cocco, O. Dubois, J. Mandler, and R. Monasson. Au seuil de la complexité combinatoire. *Pour la science*, May 2002.

[CJH03] K. Chatterjee, M. Jurdzinski, and T. Henzinger. Simple stochastic parity games. In *Proc. Int. Conf. for Computer Science Logic (CSL03)*, volume 2719 of *LNCS*, pages 886–902. Springer, 2003.

[CP00] D. Coppersmith and I. Pak. Random walk on upper triangular matrices mixes rapidly. *Probability Theory and Related Fields*, 117 :407–417, 2000.

[CY90] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proc. of 17th Int. Colloq. Aut. Lang. Prog.*, pages 336–349. LNCS, volume 443, 1990.

[dA97a] L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, Dec. 1997.

[dA97b] L. de Alfaro. Temporal logics for the specification of performance and reliability. In *Proc. of Symp. on Theor. Asp of Comp. Sci. (STACS), volume 1200 of Lect. Notes in Comp. Sci.*, pages 165–176, 1997.

[dA99] L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *Proc. of CONCUR 99.* LNCS, volume 1664, 1999.

[DEHP93] B. Derrida, M.R. Evans, V. Hakim, and V. Pasquier. Exact solution of a 1D asymmetric exclusion model using a matrix formulation. *J. Phys. A : Math Gen*, 30(3165), 1993.

[Del95] S. Delaët. *Auto-Stabilisation : Modèle et applications à l'exclusion mutuelle.* PhD thesis, Université de Paris Sud, Dec. 1995.

[Der70] C. Derman. *Finite State Markov Decision Processes.* Academic Press, 1970.

[DFH+04] M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic model-checking of the CSMA/CD protocol using PRISM and APMC. In *Proc. of 4th Int. Workshop on Automated Verification of Critical Systems (AVoCS)*, pages 336–349. ENTCS, 2004.

[DFK92] M.E. Dyer, A. Frieze, and R. Kannan. A random polymomial time algorithm for approximating the volume of a convex body. In *Proc. 24th ACM Symp. on Theory Of Computing (STOC92)*, pages 26–38. ACM, 1992.

[DFP01a] M. Duflot, L. Fribourg, and C. Picaronny. Randomized distributed algorithms as Markov chains. In *Proc. 15th Int. Conf. on Distributed Computing (DISC 2001), LNCS 2180*, pages 240–254. Springer, 2001.

[DFP01b] M. Duflot, L. Fribourg, and C. Picaronny. Randomized finite-state distributed algorithms as Markov chains. In *Proc. 15th Int. Conf. on Distributed Computing (DISC'01)*, volume 2180 of *LNCS*, pages 240–254. Springer, 2001.

[DG98] M.E. Dyer and C. Greenhill. A more rapidly mixing Markov chain for graph colorings. *Random Structures and Algorithms*, 13 :285–317, 1998.

[DGG+02] M. Dyer, L.A. Goldberg, C. Greenhill, G. Istrate, and M. Jerrum. Convergence of the Iterated Prisoner's Dilemma Game. *Combinatorics, Probability and Computing*, 11(2), 2002.

[DHT04] M. Duchon, N. Hanusse, and S. Tixeuil. Optimal randomized self-stabilizing algorithms on synchronous rings. In *Proc. of Int. Conference on Distributed Computing (DISC 2004) Amsterdam*, pages 216–229, 2004.

[Dij73] E. W. Dijkstra. Ewd 391 : Self-stabilization in spite of distributed control. In *Selected Writings on Computing : A personnal perspective*, pages 41–46. Springer-Verlag, 1973. (written in 1973, published in 1982).

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

[DKN02] C. Daws, M.Z. Kwiatkowska, and G. Norman. Automated verification of the IEEE root contention protocol with Kronos andPRISM. In *Proc. of 7th Int. Workshop on Formal Methods for Industrial Critical Systems (FMICS 2002)*, ENTCS, pages 107–122, 2002.

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

[Doo94] J.L. Doob. *Measure Theory*. Springer Verlag, 1994.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool kronos. *Hybrid Systems III*, 1066 of LNCS :208–219, 1996.

[DSC96] P. Diaconis and L. Saloff-Coste. Logarithmic Sobolev inequalities for finite Markov chains. *The Annals of Applied Probability*, 6(3) :695–750, 1996.

[Duf03] M. Duflot. *Algorithmes distribués sur des anneaux paramétrés, Preuves de convergence probabiliste et déterministe*. PhD thesis, ENS Cachan, Sep. 2003.

[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.

[Fel70] W. Feller. *An Introduction to Probability theory and its applications*. John Wiley, 1970.

[Fil91] J.A. Fill. Eigenvalue bounds on convergence to stationarity for non-reversible Markov chains. *The Annals of Applied Probability*, 1 :62–87, 1991.

[FMP03] L. Fribourg, S. Messika, and C. Picaronny. On the absence of phase transition in randomized distributed algorithms. Technical report, LSV-03-07, ENS de Cachan, France, April 2003. Available on `http ://www.lsv.ens-cachan.fr/~fribourg/publis.php`.

[FMP04a] L. Fribourg, S. Messika, and C. Picaronny. Coupling and self-stabilisation. *Selected for a special issue of Distributed Computing*, 2004.

[FMP04b] L. Fribourg, S. Messika, and C. Picaronny. Coupling and self-stabilization. In *Proc. of 18th Int. Conference on Distributed Computing (DISC2004)*, volume 3274, pages 201–215. LNCS, 2004.

[FMP04c] L. Fribourg, S. Messika, and C. Picaronny. Mixing time of the asymmetric simple exclusion process on a ring with two particles. Technical report, LSV, ENS Cachan, June 2004.

[FMP04d] L. Fribourg, S. Messika, and C. Picaronny. Parametric and probabilistic verification of Nicollin-Sifakis-Yovine's model of the CSMA-CD protocol. Technical report, LSV, ENS Cachan, June 2004. Lot 4.2 fourniture 3, du projet RNTL Averroes.

[GDO94] R.L. Graham, Knuth D.E., and Patashnik O. *Concrete Mathematics*. Addison-Wesley, 1994.

[GS92] L.H. Gva and H. Spohn. Bethe solution for the dynamical-scaling exponent of the noisy burgers equation. *Physic Review Letters*, 68(725), 1992.

[GT88] T.A. Gonsalves and F.A. Tobagi. On the performance effects of station location and access protocol parameters in ethernet networks. *IEEE Transactions on Communications*, 36(4) :441–449, 1988.

[Hag01] O. Haggstrom. A note on disagreement percolation. *Rand. Structures Algorithms*, 18 :267–278, 2001.

[Han91] H.A. Hansson. *Time and probability in formal design of distributed systems*. PhD thesis, Uppsala University, Sept. 1991.

[HHWT97] T. Henzinger, P.H. Ho, and H. Wong-Toi. Hytech : a model checker for hybrid systems. *Software tools for technology transfer*, 1 :110–122, 1997.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspect of Computing*, 6(5) :512–535, 1994.

[HLMP04] T. Herault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *Proc. of 5th Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI'04)*, volume 2937, pages 73–84. LNCS, January 2004.

[HMP92] T.A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proc. 19th International Colloquium on automata languages and rogramming (ICALP'92)*, pages 545–558. LNCS 623, 1992.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model-checking for real time systems. *Information and Computation*, 111(2) :193–244, june 1994.

[Hug04] T. Hugel. Détermination des politiques extrémales dans les processus de décision markoviens, Juillet 2004. Stage de MIM1, LSV CNRS et ENS Cachan.

[HyT] HyTech. HyTech web page. http ://www-cad.eecs.berkeley.edu/ tah/HyTech/.

[IEE] IEEE. IEEE web page. http ://www.ieee802.org/3.

[JS96] M. Jerrum and A. Sinclair. *The Markov chain Monte Carlo method : an approach to approximate counting and integration*. In "Approximation Algorithms for NP-hard Problems", D.S.Hochbaum ed., PWS Publishing, 1996.

[JSV01] M.R. Jerrum, A. Sinclair, and E. Vigoda. A polynomial time approximation algorithm for the permanent of a matrix with non-negative entries. In *Proc. 33th ACM Symp. on Theory Of Computing (STOC01)*, pages 712–721. ACM, 2001.

[Kar02] D.R. Karger. Randomization in graph optimization problems : A survey. *Optima 58*, 2002.

[Kat93] J.P. Katoen. A semi-Markov model of a home network access protocol. In *Proc. of Int. Workshop on Modeling, Analysis, and Simulation On COmputer and Telecommunication Systems*, pages 293–298, 1993.

[Kit95] J.E. Kittock. Emergent conventions and the structure of multi-agent systems. In L. Nadel and D. Stein, editors, *Proc. of the 1993 Complex systems summer school.* Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley, 1995.

[KNP02] M. Kwiatkowska, G. Norman, and D. Parker. PRISM : Probabilistic symbolic model checker. In *Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'02)*, pages 200–204. LNCS 2324, April 2002.

[KNS] M.Z. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model-checking of deadline properties in the IEEE 1394 firewire root contention protocol.

[KNS02] M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model-checking of the IEEE 802.11 wireless local area network protocol. In *Proc. of the 2nd International Workshop PAPM-PROBMIV*, pages 169–187. LNCS 2399, July 2002.

[KNSS02] M. Kwiatkowska, G. Norman, R. Segala, and J Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical computer science*, 282(1) :101–150, 2002.

[KRO] note = http ://www-verimag.imag.fr/TEMPORISE/kronos/ KRONOS, title = KRONOS web page.

[KSK66] J. G. Kemeny, J. L. Snell, and A. W. Knapp. *Denumerable Markov Chains.* D. van Nostrand Co., 1966.

[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.

[LRS95] M. Luby, D. Randall, and A. Sinclair. Markov chain algorithms for planar lattice structures (extended abstract). In *Proc. of the 36th Annual IEEE Symp. on Foundations of Computer Science (FOCS'95)*, pages 150–159, 1995.

[LSP82] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3) :382–401, 1982.

[LW98] L. Lovász and P. Winkler. Mixing Times. *Microsurveys in Discrete Probability*, pages 85–134, 1998.

[Lyn96] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, 1996.

[MK00] S.N. Majumdar and P.L. Krapivsky. Extremal paths on a random Cayley tree. *Phys. Rev. E.*, 62(7735), 2000.

[Mol02] M. Molloy. The Glauber dynamics on colorings of a graph with high girth and maximum degree. In *Proc. 34th ACM Symp. on Theory Of Computing (STOC02)*, pages 91–98. ACM, 2002.

[MOY96] A. Mayer, R. Ostrovsky, and M. Yung. Self-Stabilizing Algorithms for Synchronous Unidirectional Rings. In *Proc. of the 7th ACM-SIAM Symp. on Discrete Algorithms (SODA-96)*, 1996.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[Nor98] J. R. Norris. *Markov Chains.* Cambridge University press, 1998.

[NSY92] X. Nicollin, J. Sifakis, and S. Yovine. Compiling real time specifications into extended automata. *IEEE Transactions on Software Engineering*, 18(9) :794–804, Sept. 1992.

[Pan01] P. Panangaden. Measure and probability for concurrency theorists. *Theoretical Computer Science*, 253(2) :287–309, 2001.

[PRI] PRISM. PRISM web page. http ://www.cs.bham.ac.uk/ dxp/prism/.

[Pri03] V.B. Priezzhev. Exact non-stationary probabilities in the asymmetric exclusion process on a ring. *Physical Review Letters*, 91(5), 2003.

[Put94] M. L. Puterman. *Markov Decision Processes : Discrete Stochastic Dynamic Programming.* Wiley-Interscience, 1994.

[PZ86] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1) :53–72, Jan. 1986.

[Ran03] D. Randall. Mixing. In *Proc. of the 44th Annual IEEE Symp. on Foundations of Computer Science (FOCS'03)*, 2003.

[RL94] M. O. Rabin and Daniel J. Lehmann. *The advantages of free choice : a symetric and fully distributed solution for the dining philosophers problem. In "A Classical Mind : Essays in Honour of C.A.R. Hoare"*, chapter 20, pages 333–352. Prentice Hall, 1994.

[Sch93] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25 :45–67, 1993.

[Sch97] G.M. Schütz. Exact solution of a the master equation for the asymmetric exclusion process. *J. Stat. Phys.*, 88(1-2) :427–445, 1997.

[Sin93] A. Sinclair. Algorithms for random generation and counting. Birkhauser, 1993. Boston.

[Sin97] A. Sinclair. Convergence rates for Monte Carlo experiments. In *Numerical Methods for Polymeric Systems*, pages 1–18. IMA Volumes in Mathematics & Its application, 1997.

[SL95] R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2) :250–273, 1995.

[Tel91] G. Tel. *Topics in Distributed Algorithms.* Cambridge University Press, 1991.

[Tel94] G. Tel. *Introduction to Distributed Algorithms.* Cambridge University Press, 1994.

[TK85] H. Takagi and L. Kleinrock. Throughput analysis for persistent CSMA system. *IEEE Transactions on Communications*, 33(7) :627–638, 1985.

[UPP] UPPAAL. UPPAAL web page. http ://www.uppaal.com/.

[Var85] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th Annual Symp. on Foundations of Computer Science(FOCS'85)*, pages 327–338. IEEE Comp. Soc. Press, 1985.

[vdB93] J. van den Berg. A uniqueness condition for Gibbs measure with application to the 2-dimensional Ising antiferromagnet. *Commun. Math. Phys.*, 152 :161–166, 1993.

[vdB99] J. van den Berg. On the absence of phase transition in the monomer-dimer model. *Perplexing Problems in Probability (Festschrift in honor of Harry Kesten), M. Branson and R. Durrett eds.*, pages 185–195, 1999.

[vdBM94] J. van den Berg and C. Maes. Disagreement percolation in the study of Markov field. *Ann. Probab.*, 22 :749–763, 1994.

[Wil91] D. Williams. *Probability with Martingales.* Cambridge University Press, 1991.

[WK99] J. Wang and S. Keshav. Efficient and accurate Ethernet simulation. In *Proc. of the IEEE 24th Conference on Local Computer Network)*, pages 1820–191, 1999.

Index