SPIM

Habilitation à Diriger des Recherches

UFC

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE FRANCHE-COMTÉ

Modèles discrets pour la sécurité informatique: des méthodes itératives à l'analyse vectorielle

JEAN-FRANÇOIS COUCHOT

SPIM

Habilitation à Diriger des Recherches

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

HABILITATION À DIRIGER DES RECHERCHES

de l'Université de Franche-Comté

préparée au sein de l'Université Bourgogne Franche-Comté

Spécialité : **Informatique**

présentée par

JEAN-FRANÇOIS COUCHOT

Modèles discrets pour la sécurité informatique: des méthodes itératives à l'analyse vectorielle

Soutenue publiquement le 30 Janvier 2017 devant le Jury composé de :

OLIVIER BOURNEZ	Rapporteur	Professeur à l'Ecole Polytechnique
JEAN-PAUL COMET	Rapporteur	Professeur à l'Université de Nice Sophia Antipolis
JUAN-PABLO ORTEGA	Rapporteur	Professeur à l'Université de St. Gallen–Suisse
SYLVAIN CONTASSOT-VIVIER	Examinateur	Professeur à l'Université de Lorraine
RAPHAËL COUTURIER	Examinateur	Professeur à l'Université de Bourgogne Franche-Comté
CHRISTOPHE GUYEUX	Examinateur	Professeur à l'Université de Bourgogne Franche-Comté

REMERCIEMENTS

Ce manuscrit est le fruit d'un travail personnel. Il a cependant été soutenu par de nombreuses personnes qui méritent beaucoup plus que les quelques lignes qui suivent en guise de témoignage de reconnaissance.

Deux personnes m'ont professionnellement guidé et doivent tout d'abord être nommées : Christophe GUYEUX et Raphaël COUTURIER. Ils se caractérisent tous les deux par une passion sans faille pour la Recherche (avec une majuscule, au sens noble) et une volonté de s'attaquer toujours à des problèmes difficiles pour *in fine* obtenir de vraies contributions. Ils m'ont transmis beaucoup de cette énergie nécessaire pour réaliser cette synthèse scientifique. J'ai eu beaucoup de chance de les avoir comme collègues.

Jacques BAHI, à l'origine de nombreux travaux présentés dans ce document, doit aussi être largement remercié. De part ses connaissances scientifiques sur les systèmes dynamiques discrets, il a été extrêmement efficace à me convaincre en 2007 de travailler sur ce sujet, alors que la déduction logique était ma spécialité. C'est un autre pan des mathématiques discrètes qui s'est ouverte à moi grâce à lui.

Je suis infiniment reconnaissant envers les trois rapporteurs de cette Habilitation à Diriger les Recherches, Olivier BOURNEZ, Jean-Paul COMET et Juan-Pablo ORTEGA. La lecture en profondeur qu'ils ont réalisée est remarquable et mérite bien plus que mes remerciements. Leurs suggestions ont ainsi pu faire progresser le document.

Beaucoup des contributions présentées dans ce manuscrit sont directement ou indirectement liées à Sylvain CONTASSOT-VIVIER, membre du jury. Il a su m'accompagner et me donner de nombreuses pistes sur un de ses domaines de recherche, les systèmes dynamiques discrets. Ces quelques lignes de remerciement ne sont rien par rapport à ce que ce document lui doit.

Alain GIORGETTI et Françoise BELLEGARDE co-directeurs de ma thèse (soutenue en 2006) et amis ont aussi leur part de responsabilité dans ce travail. Il m'ont appris la rigueur, le recul, la justesse scientifique. Si ces quelques-uns de ces travaux ont une valeur scientifique, c'est en partie grâce à eux.

Que tous les co-auteurs cités dans ce manuscrits méritent également ma reconnaissance à savoir : Bassam ALKINDY, Bashar AL-NUAIMI, Jacques BAHI, Mohamed BAKIRI, Ahmad BITTAR, Sylvain CONTASSOT-VIVIER, Raphaël COUTURIER, Karine DESCHINKEL, Rony DARAZI, Yousra Ahmed FADIL, Nicolas FRIOT, Christophe GUYEUX, Pierre-Cyrille HÉAM, Kamel MAZOUZI, Ahmed MOSTEFAOUI, Adrien RICHARD, Laurent PHILIPPE, Michel SALOMON et Qianxue WANG. Chaque rédaction d'article avec chacun d'eux a toujours été un moment de plaisir.

Je remercie enfin celle qui a supporté ceci sans jamais plier : Sylvaine, ma chérie. Humeurs noires, doutes, épuisement, absences même lorsque j'étais physiquement présent. Tout y est passé. Ses encouragements, son relativisme, sa joie m'ont aidé beaucoup plus qu'elle ne veut l'admettre. Ces remerciements s'arrêtent sur elle comme lors-

qu’on garde le meilleur pour la fin.

SOMMAIRE

INTRODUCTION

MOTIVATIONS

Les informations traitées par un ordinateur ne sont, *in fine*, que discrètes : les flottants (sur un nombre fini de bits) sont une interprétation des réels, les *longs* une interprétation finie des entiers... Les phénomènes physiques ou naturels peuvent aussi être modélisés par des approches discrètes : il n'est parfois pas nécessaire de suivre exactement tous les états par lesquels peut passer l'élément d'étude. Pour peu que l'on sache identifier les seuils de son domaine (c'est-à-dire les valeurs critiques permettant au système global de changer), il est parfois suffisant de réduire le domaine de l'état de l'élément à un ensemble fini d'intervalles définis par ces seuils. Un élément passe d'un état i à un état j si dans la version continue il passe d'un état de l'intervalle numéro i à un état de l'intervalle numéro j. En abstrayant à l'extrême, on peut considérer pour chaque élément seulement deux états : 1 pour le fait d'être présent et 0 pour le fait d'être absent, ou bien 1 pour le fait d'avoir une valeur supérieure à un seuil et 0 pour le fait qu'elle y soit inférieure. Ces réseaux particuliers sont qualifiés de *réseaux booléens*.

Soit N un entier naturel représentant le nombre d'éléments étudiés (le nombre de bits qu'un générateur veut engendrer, le nombre de pixels que l'on veut modifier...) un réseau booléen est défini à partir d'une fonction booléenne $f : \mathbb{B}^N \to \mathbb{B}^N$ et un mode de mise à jour. A chaque étape, on peut intuitivement ne modifier qu'une partie des N éléments. En fonction des éléments qui sont susceptibles d'être modifiés à chaque étape, on peut se poser les questions de savoir si le réseau atteint un point fixe (et qu'il peut donc converger) ou s'il possède des attracteurs cycliques (et qu'il diverge donc).

Détecter la convergence ou son contraire peut se faire *a priori* dans certains cas. En effet, de nombreux travaux ont énoncé des conditions suffisantes sur les réseaux booléens garantissant leur convergence ou leur divergence. Lorsqu'aucune d'entre elles ne s'applique, on peut penser à étendre ces résultats théoriques et compléter ceci par des simulations. Lorsque la convergence est pratiquement observée, il reste à vérifier que celle-ci est indépendante du choix des parties à modifier, par exemple. Une vérification *a posteriori* s'impose, sauf si la méthode de simulation a été prouvée comme exhaustive.

Une fonction qui admet un attracteur cyclique égal à l'ensemble des sommets diverge. Admet-il cependant un comportement chaotique ? Les théories mathématiques du chaos ont énoncé des critères permettant de décider si le comportement d'une fonction est chaotique ou non, et plus récemment si certains réseaux booléens l'étaient. Se pose légitimement la question de savoir si cette caractérisation s'étend quels que soient les parties modifiées à chaque étape. Naturellement, ceci n'a un sens pratique que si le nombre de réseaux booléens qui possèdent cette caractéristique est suffisamment grand et que l'on sait engendrer algorithmiquement de tels réseaux.

Les liens entre chaos et aléas sont certes intuitifs, mais sont-ils suffisants pour espérer construire un générateur de nombres pseudo-aléatoires (PRNG) à partir de fonctions

au comportement chaotique ? La réponse est certes positive, mais pas triviale. Il est nécessaire de définir les propriétés attendues de la sortie d'un PRNG et l'uniformité de sa distribution est sans doute la première des propriétés à assurer. Comment alors générer des fonctions chaotiques dont les itérations peuvent converger vers une distribution uniforme ? Combien d'itérations suffit-il alors d'effectuer pour s'approcher suffisamment de celle-ci ?

Les liens entre chaos et marquage d'information sont aussi faciles à établir : de tout média marqué même attaqué, la marque doit pouvoir être extraite. Cette propriété du marquage est proche en effet de celle de régularité des opérateurs chaotiques. Il est alors naturel d'envisager exploiter les fonctions chaotiques extraites dans ce contexte et donc de modifier certains bits d'un média pour y insérer de l'information : la marque.

Les compétences acquises dans l'étude des algorithmes d'insertion d'une marque dans une image nous permettent aussi d'adresser le problème d'insérer un message dans une image. Cependant, il s'agit de privilégier cette fois l'imperceptibilité et non plus la robustesse. Ainsi, tandis que l'idée principale était d'étaler le message sur un ensemble conséquent de pixels pour garantir la robustesse, il s'agit ici de sélectionner finement ceux dont les modifications seraient le moins perceptibles possible. On pense immédiatement à insérer ces messages dans les pixels contenant les zones les plus perturbées. Les outils mathématiques d'analyse permettant d'identifier les lignes de niveaux pour ensuite voir lesquelles sont les moins régulières (les plus perturbées) sont le gradient et la matrice Hessienne. Cependant, ces modèles d'analyse ne sont définis que pour des fonctions de $\mathbb{R}^n$ dans $\mathbb{R}$. Se pose alors la question sur la possibilité de les adapter au cadre discret puisque les images à traiter sont construites à partir de pixels dont les valeurs sont discrètes.

ORGANISATION DE CE MÉMOIRE

Ce mémoire est organisé en quatre parties.

La première partie sur les réseaux discrets. Dans celle-ci, le chapitre 1 formalise la notion de réseaux booléens et leurs modes opératoires. On y définit notamment un nouveau mode opératoire assurant la convergence et ce en un temps réduit. Les résultats de convergence suffisants à la compréhension de ce mémoire y sont rappelés et ceux relatifs à ce nouveau mode y sont prouvés.

Le chapitre 2 montre comment nous avons développé une démarche de preuve automatique de convergence de réseaux discrets. La démarche est prouvée correcte et complète : le verdict donné par l'outil est celui qui serait donné par une preuve mathématique.

La seconde partie établit globalement le lien entre les réseaux discrets et le chaos. Le chapitre 3 rappelle tout d'abord les notions suffisantes concernant la théorie du chaos. Une caractérisation des fonctions engendrant des itérations chaotiques y est donnée, selon le mode unaire et le mode généralisé, les deux modes au cœur de ce mémoire. Un théorème, démontré, propose un algorithme permettant de générer des fonctions possédant cette caractéristique.

Les itérations unaires de telles fonctions étant chaotiques, nous avons étudié au chapitre 4 la possibilité de les faire apprendre par un réseau de neurones, plus précisement

par un perceptron multi-couches.

Le titre de la troisième partie donne une idée de la conclusion de cette étude puisqu'on y étudie une famille PRNG construite à partir de fonctions dont les itérations sont chaotiques. Plus précisément, le chapitre 5 caractérise les PRNG construits à partir de réseaux booléens qui sont chaotiques en donnant des conditions suffisantes sur la fonction à itérer. Le chapitre 6 s'intéresse donc à générer ce type de fonction de manière autrement plus efficace qu'à partir de la méthode décrite au chapitre 3. On y présente aussi un majorant du nombre d'itérations à effectuer pour obtenir une distribution uniforme.

Comme annoncé dans les motivations à ce travail, les itérations chaotiques peuvent s'appliquer au marquage de média et plus généralement au masquage d'information. C'est l'objectif de la quatrième partie.

Dans le premier chapitre de celle-ci (chapitre 7), nous formalisons le processus de marquage d'information. Grâce à cette formalisation, nous pouvons étudier des propriétés de stégo-securité et chaos-sécurité.

Les deux chapitres suivants (chapitre 8 et chapitre 9) sont une parenthèse au domaine discret puisqu'on s'intéresse au marquage de document PDF par une méthode classique et au masquage d'information par une technique de détection de bords.

Le dernier chapitre des contributions (chapitre 10) retourne dans le monde discret. Il montre qu'on peut approximer efficacement à l'aide de matrices discrètes des calculs de gradients pour, *in fine*, construire des lignes de niveau et embarquer de l'information dans les lignes de niveau les moins régulières.

Une conclusion et des perspectives sont données en dernière partie.

PUBLICATIONS EN TANT QU'ENSEIGNANT-CHERCHEUR

Le tableau de la figure 1 donné ci dessous synthétise les références auxquelles j'ai participé depuis mon intégration en tant qu'enseignant chercheur.

Journaux internationaux	Conférences internationales
[BCG12a, BCG12b, BCGS12] [CDS13, CCG15, BDCC15] [CCVHG16]	[AAG+15, BCFG12a, BCFG12b, BCF+13, BCG11a] [BCG11b, ACGS13, CHG+14b] [BCGR11, BCGW11, CDS12, CHG+14a, FCCG15] [BCC+15, BCG16, CCFG16, NK16]

FIGURE 1 – Bilan synthétique des publications

I

RÉSEAUX DISCRETS

1

ITERATIONS DISCRÈTES DE RÉSEAUX BOOLÉENS

Ce chapitre formalise tout d'abord ce qu'est un réseau booléen (section 1.1. On y revoit les différents modes opératoires, leur représentation à l'aide de graphes et les résultats connus de convergence). Ce chapitre montre ensuite à la section 1.2 comment combiner ces modes pour converger aussi souvent, mais plus rapidement vers un point fixe. Les deux dernières sections ont fait l'objet du rapport [BCVC10].

1.1/ FORMALISATION

On considère l'espace booléen $\mathbb{B} = \{0, 1\}$ muni des opérateurs binaires de disjonction « + », de conjonction « . » et unaire de négation « $\overline{\ }$ ».

Soit N un entier naturel. On introduit quelques notations à propos d'éléments de $\mathbb{B}^N$. L'ensemble $\{1, \ldots, N\}$ sera par la suite noté [N]. Le $i^{\text{ème}}$ composant d'un élément $x \in \mathbb{B}^N$ s'écrit x_i. Si l'ensemble I est une partie de [N], alors $\overline{x}^I$ est l'élément $y \in \mathbb{B}^N$ tel que $y_i = 1 - x_i$ si $i \in I$ et $y_i = x_i$ sinon. On considère les deux abréviations $\overline{x}$ pour $\overline{x}^{[N]}$ (chaque composant de $\overline{x}$ est nié : c'est une négation composante à composante) et $\overline{x}^i$ pour $\overline{x}^{\{i\}}$ pour $i \in$ [N] (seul x_i est nié dans $\overline{x}$). Pour tout x et y dans $\mathbb{B}^N$, l'ensemble $\Delta(x, y)$, contient les $i \in$ [N] tels que $x_i \neq y_i$. Soit enfin $f : \mathbb{B}^N \to \mathbb{B}^N$. Son $i^{\text{ème}}$ composant est nommé f_i qui est une fonction de $\mathbb{B}^N$ dans $\mathbb{B}$. Pour chaque x dans $\mathbb{B}^N$, l'ensemble $\Delta f(x)$ est défini par $\Delta f(x) = \Delta(x, f(x))$. On peut admettre que $f(x) = \overline{x}^{\Delta f(x)}$.

Exemple. *On considère* N = 3 *et* $f : \mathbb{B}^3 \to \mathbb{B}^3$ *telle que* $f(x) = (f_1(x), f_2(x), f_3(x))$ *avec*

$$\begin{aligned} f_1(x_1, x_2, x_3) &= (\overline{x_1} + \overline{x_2}).x_3, \\ f_2(x_1, x_2, x_3) &= x_1.x_3 \ et \\ f_3(x_1, x_2, x_3) &= x_1 + x_2 + x_3. \end{aligned}$$

La TABLE *1.1 donne l'image de chaque élément* $x \in \mathbb{B}^3$*. Pour* $x = (0, 1, 0)$ *les assertions suivantes se déduisent directement du tableau :*

- $f(x) = (0, 0, 1)$ *;*
- *pour* $I = \{1, 3\}$, $\overline{x}^I = (1, 1, 1)$ *et* $\overline{x} = (1, 0, 1)$ *;*
- $\Delta(x, f(x)) = \{2, 3\}$.

x			$f(x)$		
x_1	x_2	x_3	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	1	0	1	1

TABLE 1.1 – Images de $(x_1, x_2, x_3) \mapsto ((\overline{x_1} + \overline{x_2}).x_3, x_1.x_3, x_1 + x_2 + x_3)$

1.1.1/ RÉSEAU BOOLÉEN

Soit N un entier naturel représentant le nombre d'éléments étudiés. Un réseau booléen est défini à partir d'une fonction booléenne :

$$f : \mathbb{B}^{\mathrm{N}} \to \mathbb{B}^{\mathrm{N}}, \qquad x = (x_1, \ldots, x_{\mathrm{N}}) \mapsto f(x) = (f_1(x), \ldots, f_{\mathrm{N}}(x)),$$

et un *schéma itératif* ou encore *mode de mise à jour*. À partir d'une configuration initiale $x^0 \in \mathbb{B}^{\mathrm{N}}$, la suite $(x^t)^{t\in\mathbb{N}}$ des configurations du système est construite selon l'un des schémas suivants :

— **Schéma parallèle synchrone :** basé sur la relation de récurrence $x^{t+1} = f(x^t)$. Tous les x_i, $1 \leq i \leq \mathrm{N}$, sont ainsi mis à jour à chaque itération en utilisant l'état global précédent du système x^t.

— **Schéma unaire :** ce schéma est parfois qualifié de chaotique dans la littérature. Il consiste à modifier la valeur d'un unique élément i, $1 \leq i \leq \mathrm{N}$, à chaque itération. Le choix de l'élément qui est modifié à chaque itération est défini par une suite $S = (s^t)^{t\in\mathbb{N}}$ qui est une séquence d'indices dans [N]. Cette suite est appelée *stratégie unaire*. Ce mode est défini pour tout $i \in$ [N] par

$$x_i^{t+1} = \begin{cases} f_i(x^t) \text{ si } i = s^t, \\ x_i^t \text{ sinon.} \end{cases} \tag{1.1}$$

— **Schéma généralisé :** dans ce schéma, ce sont les valeurs d'un ensemble d'éléments de [N] qui sont modifiées à chaque itération. Dans le cas particulier où c'est la valeur d'un singleton $\{k\}$, $1 \leq k \leq \mathrm{N}$, qui est modifiée à chaque itération, on retrouve le mode unaire. Dans le second cas particulier où ce sont les valeurs de tous les éléments de $\{1, \ldots, \mathrm{N}\}$ qui sont modifiées à chaque itération, on retrouve le mode parallèle. Ce mode généralise donc les deux modes précédents. Plus formellement, à la $t^{\text{ème}}$ itération, seuls les éléments de la partie $s^t \in \mathcal{P}([\mathrm{N}])$ sont mis à jour. La suite $S = (s^t)^{t\in\mathbb{N}}$ est une séquence de sous-ensembles de [N] appelée *stratégie généralisée*. Ce schéma est basé sur la relation définie pour tout $i \in$ [N] par

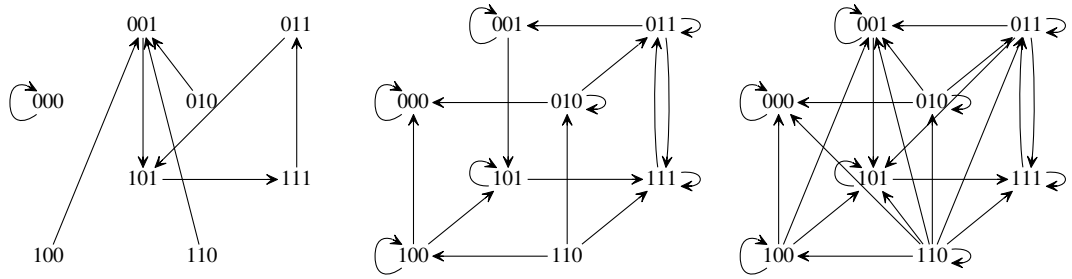
$$x_i^{t+1} = \begin{cases} f_i(x^t) \text{ si } i \in s^t, \\ x_i^t \text{ sinon.} \end{cases} \tag{1.2}$$

La section suivante détaille comment représenter graphiquement les évolutions de tels réseaux.

1.1.2/ GRAPHES DES ITÉRATIONS

Pour un entier naturel N et une fonction $f : B^N \to B^N$, plusieurs évolutions sont possibles en fonction du schéma itératif retenu. Celles-ci sont représentées par un graphe orienté dont les noeuds sont les éléments de $\mathbb{B}^N$ (voir FIGURE 1.1).

— Le *graphe des itérations synchrones* de f, noté GIS(f) est le graphe orienté de $\mathbb{B}^N$ qui contient un arc $x \to y$ si et seulement si $y = f(x)$.
— Le *graphe des itérations unaires* de f, noté GIU(f) est le graphe orienté de $\mathbb{B}^N$ qui contient un arc $x \to y$ si et seulement s'il existe $i \in \Delta f(x)$ tel que $y = \overline{x}^i$. Si $\Delta f(x)$ est vide, on ajoute l'arc $x \to x$.
— Le *graphe des itérations généralisées* de f, noté GIG(f) est le graphe orienté de $\mathbb{B}^N$ qui contient un arc $x \to y$ si et seulement s'il existe un ensemble $I \subseteq \Delta f(x)$ tel que $y = \overline{x}^I$. On peut remarquer que ce graphe contient comme sous-graphe à la fois celui des itérations synchrones et celui des itérations unaires.



(a) GIS(f) (b) GIU(f) (c) GIG(f)

FIGURE 1.1 – Graphes des itérations de la fonction $f : \mathbb{B}^3 \to \mathbb{B}^3$ telle que $(x_1, x_2, x_3) \mapsto ((\overline{x_1} + \overline{x_2}).x_3, x_1.x_3, x_1 + x_2 + x_3)$. On remarque le cycle $((101, 111), (111, 011), (011, 101))$ à la FIGURE (1.1(a)).

Exemple. *On reprend notre exemple illustratif détaillé à la page 3 avec sa table d'images (*TABLE *1.1). La* FIGURE *1.1 donne les trois graphes d'itérations associés à f.*

1.1.3/ ATTRACTEURS

On dit que le point $x \in \mathbb{B}^N$ est un *point fixe* de f si $x = f(x)$. Les points fixes sont particulièrement intéressants car ils correspondent aux états stables : dans chaque graphe d'itérations, le point x est un point fixe si et seulement si il est son seul successeur.

Soit un graphe d'itérations (synchrones, unaires ou généralisées) de f. Les *attracteurs* de ce graphe sont les plus petits sous-ensembles (au sens de l'inclusion) non vides $A \subseteq \mathbb{B}^N$ tels que pour tout arc $x \to y$, si x est un élément de A, alors y aussi. Un attracteur qui contient au moins deux éléments est dit *cyclique*. On en déduit qu'un attracteur cyclique ne contient pas de point fixe. En d'autres termes, lorsqu'un système entre dans un attracteur cyclique, il ne peut pas atteindre un point fixe.

On a la proposition suivante :

Théorème [1]. *La configuration x est un point fixe si et seulement si $\{x\}$ est un attracteur du graphe d'itérations (synchrone, unaire, généralisé). En d'autres termes, les attracteurs non cycliques de celui-ci sont les points fixes de f. Ainsi pour chaque $x \in \mathbb{B}^N$, il existe au moins un chemin depuis x qui atteint un attracteur. Tout graphe d'itérations contient donc toujours au moins un attracteur.*

Exemple. *Les attracteurs de* GIU(f) *et de* GIG(f) *sont le point fixe* 000 *et l'attracteur cyclique* $\{001, 101, 111, 011\}$. *Les attracteurs de* GIS(f) *sont le point fixe* 000 *et l'attracteur cyclique* $\{011, 101, 111\}$.

1.1.4/ GRAPHE D'INTERACTION

Les interactions entre les composants du système peuvent être mémorisées dans la *matrice jacobienne discrète* f'. Celle-ci est définie comme étant la fonction qui à chaque configuration $x \in \mathbb{B}^N$ associe la matrice de taille $n \times n$ telle que

$$f'(x) = (f'_{ij}(x)), \qquad f'_{ij}(x) = \frac{f_i(\overline{x}^j) - f_i(x)}{\overline{x_j} - x_j} \qquad (i, j \in [n]). \tag{1.3}$$

On note que dans l'équation donnant la valeur de $f'_{ij}(x)$, les termes $f_i(\overline{x}^j)$, $f_i(x)$, $\overline{x}^j_j$ et x_j sont considérés comme des entiers naturels égaux à 0 ou à 1 et que la différence est effectuée dans $\mathbb{Z}$. Lorsqu'on supprime les signes dans la matrice jacobienne discrète, on obtient une matrice notée $B(f)$ aussi de taille N × N. Celle-ci mémorise uniquement l'existence d'une dépendance de tel élément vis à vis de tel élément. Elle ne mémorise pas *comment* les éléments dépendent les uns par rapport aux autres. Cette matrice est nommée *matrice d'incidence*.

Théorème [2]. *Si f_i ne dépend pas de x_j, alors pour tout $x \in$ [N], $f_i(\overline{x}^j)$ est égal à $f_i(x)$, i.e., $f'_{ij}(x) = 0$. Ainsi $B(f)_{ij}$ est nulle. La réciproque est aussi vraie.*

En outre, les interactions peuvent se représenter à l'aide d'un graphe $\Gamma(f)$ orienté et signé défini ainsi : l'ensemble des sommets est [N] et il existe un arc de j à i de signe $s \in \{-1, 1\}$, noté (j, s, i), si $f_{ij}(x) = s$ pour au moins un $x \in \mathbb{B}^N$.

On note que la présence de deux arcs de signes opposés entre deux sommets donnés est possible. Un cycle *positif* (resp. négatif) de G est un cycle *élémentaire* qui contient un nombre pair (resp. impair) d'arcs négatifs. La *longueur* d'un cycle est le nombre d'arcs qu'il contient.

Exemple. *Pour exprimer la jacobienne discrète de la fonction donnée en exemple, pour chaque i, j dans* $[3]$ *on exprime $f'_{ij}(x) = \frac{f_i(\overline{x}^j) - f_i(x)}{\overline{x_j} - x_j}$ d'après l'équation (1.3). La* FIGURE *(1.2(a)) explicite la matrice jacobienne discrète de cette fonction.*

Le graphe d'interaction de f s'en déduit directement. Il est donné à la FIGURE *(1.2(c)). Il possède un cycle négatif de longueur 1 et un cycle négatif de longueur 3. Concernant les cycles positifs, il en possède un de longueur 1, deux de longueur 2 et un de longueur 3.*

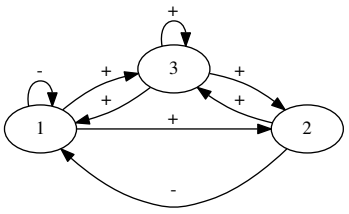
La matrice d'incidence peut se déduire de la matrice d'interaction en supprimant les signes ou bien en constatant que f_1 dépend des trois éléments x_1, x_2 et x_3 et donc que la première ligne de $B(f)$ est égale à $1\ 1\ 1$. *De manière similaire, f_2 (resp. f_3) dépend de x_1 et de x_3 (resp. dépend de x_1, x_2 et x_3). Ainsi la seconde ligne (resp. la troisième ligne) de $B(f)$ est* $1\ 0\ 1$ *(resp. est* $1\ 1\ 1$*). La* FIGURE *(1.2(b)) donne la matrice d'incidence complète.*

$$\begin{pmatrix} \frac{((x_1+\overline{x_2}).x_3)-((\overline{x_1}+\overline{x_2}).x_3)}{\overline{x_1}-x_1} & \frac{((\overline{x_1}+x_2).x_3)-((\overline{x_1}+\overline{x_2}).x_3)}{\overline{x_2}-x_2} & \frac{((\overline{x_1}+\overline{x_2}).\overline{x_3})-((\overline{x_1}+\overline{x_2}).x_3)}{\overline{x_3}-x_3} \\ \frac{\overline{x_1}.x_3-x_1.x_3}{\overline{x_1}-x_1} & 0 & \frac{x_1.\overline{x_3}-x_1.x_3}{\overline{x_3}-x_3} \\ \frac{(\overline{x_1}+x_2+x_3)-(x_1+x_2+x_3)}{\overline{x_1}-x_1} & \frac{(x_1+\overline{x_2}+x_3)-(x_1+x_2+x_3)}{\overline{x_2}-x_2} & \frac{(x_1+x_2+\overline{x_3})-(x_1+x_2+x_3)}{\overline{x_3}-x_3} \end{pmatrix}$$

(a) Matrice jacobienne

$$B(f) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

(b) Matrice d'incidence



(c) Graphe d'interaction

FIGURE 1.2 – Représentations des dépendances entre les éléments de la fonction f : $\mathbb{B}^3 \to \mathbb{B}^3$ telle que $(x_1, x_2, x_3) \mapsto ((\overline{x_1} + \overline{x_2}).x_3, x_1.x_3, x_1 + x_2 + x_3)$

Soit P une suite d'arcs de $\Gamma(f)$ de la forme

$$(i_1, s_1, i_2), (i_2, s_2, i_3), \ldots, (i_r, s_r, i_{r+1}).$$

Alors, P est dit un chemin de $\Gamma(f)$ de longueur r et de signe $\Pi_{i=1}^{r} s_i$ et i_{r+1} est dit accessible depuis i_1. P est un *circuit* si $i_{r+1} = i_1$ et si les sommets $i_1, \ldots i_r$ sont deux à deux disjoints. Un sommet i de $\Gamma(f)$ a une *boucle* positive (resp. négative) , si $\Gamma(f)$ a un arc positif (resp. un arc négatif) de i vers lui-même.

1.1.5/ CONDITIONS DE CONVERGENCE

Parmi les itérations unaires caractérisées par leurs stratégies $S = (s^t)^{t\in\mathbb{N}}$ d'éléments appartenant à [N], sont jugées intéressantes celles qui activent au moins une fois chacun des $i \in$ [N]. Dans le cas contraire, un élément n'est jamais modifié.

Plus formellement, une séquence finie $S = (s^t)^{t\in\mathbb{N}}$ est dite *complète* relativement à [N] si tout indice de [N] s'y retrouve au moins une fois.

Parmi toutes les stratégies unaires de $[N]^{\mathbb{N}}$, on qualifie de :

— *périodiques* celles qui sont constituées par une répétition infinie d'une même séquence S complète relativement à [N]. En particulier toute séquence périodique est complète.

— *pseudo-périodiques* celles qui sont constituées par une succession infinie de séquences (de longueur éventuellement variable non supposée bornée) complètes. Autrement dit dans chaque stratégie pseudo-périodique, chaque indice de 1 à N revient infiniment.

On a le théorème suivant de convergence dans le mode des itérations unaires.

Théorème 3 ([Rob95]). *Si le graphe $\Gamma(f)$ n'a pas de cycle et si la stratégie unaire est*

pseudo-périodique, alors tout chemin de GIU(f) *atteint l'unique point fixe* ζ *en au plus* N *pseudo-périodes.*

Le qualificatif *pseudo-périodique* peut aisément s'étendre aux stratégies généralisées comme suit. Lorsqu'une stratégie généralisée est constituée d'une succession infinie de séquences de parties de [N] dont l'union est [N], cette stratégie est pseudo-périodique. On a le théorème suivant.

Théorème [4] ([Bah00])**.** *Si le graphe* $\Gamma(f)$ *n'a pas de cycle et si la stratégie généralisée est pseudo-périodique alors tout chemin de* GIG(f) *(et donc de* GIU(f)*) finit par atteindre l'unique point fixe* ζ*.*

1.2/ COMBINAISONS SYNCHRONES ET ASYNCHRONES

Pour être exécuté, le mode des itérations généralisées nécessite que chaque élément connaisse la valeur de chaque autre élément dont il dépend. Pratiquement, cela se réalise en diffusant les valeurs des éléments de proche en proche à tous les composants avant chaque itération. Dans le mode généralisé *asynchrone*, le composant n'attend pas : il met à jour sa valeur avec les dernières valeurs dont il dispose, même si celles-ci ne sont pas à jour. Cette section vise l'étude de ce mode.

Pratiquement, chaque stratégie du mode généralisé peut être mémorisée comme un nombre décimal dont la représentation en binaire donne la liste des éléments modifiés. Par exemple, pour un système à 5 éléments la stratégie définie par

$$s^t = 24 \text{ si } t \text{ est pair et } s^t = 15 \text{ sinon} \tag{1.4}$$

active successivement les deux premiers éléments (24 est 11000) et les quatre derniers éléments (15 est 01111). Dans le mode asynchrone, à chaque itération t, chaque composant peut mettre à jour son état en fonction des dernières valeurs qu'il connaît des autres composants. Obtenir ou non les valeurs les plus à jour dépend du temps de calcul et du temps d'acheminement de celles-ci. On parle de latence, de délai.

Formalisons le mode des itérations asynchrone. Soit $x^0 = (x_1^0, \ldots, x_n^0)$ une configuration initiale. Soit $(D^t)^{t\in\mathbb{N}}$ la suite de matrices de taille $n \times n$ dont chaque élément D_{ij}^t représente la date (inférieure ou égale à t) à laquelle la valeur x_j produite par le composant j devient disponible au composant i. On considère que le délai entre l'émission par j et la réception par i, défini par $\delta_{ij}^t = t - D_{ij}^t$, est borné par une constante δ_0 pour tous les i, j. Le *mode des itérations généralisées asynchrone* est défini pour chaque $i \in \{1, \ldots, n\}$ et chaque $t = 0, 1, 2, \ldots$ par :

$$x_i^{t+1} = \begin{cases} f_i(x_1^{D_{i1}^t}, \ldots, x_n^{D_{in}^t}) \text{ si } i \in s^t \\ x_i^t \text{ sinon.} \end{cases} \tag{1.5}$$

où *bin* convertit un entier en un nombre binaire. Les itérations de f sont *convergentes* modulo une configuration initiale x^0, une stratégie s et une matrice de dates $(D^t)^{t\in\mathbb{N}}$, si la fonction atteint un point fixe. Cela revient à vérifier la propriété suivante :

$$\exists t_0 \,.\, (\forall t \,.\, t \geq t_0 \Rightarrow x^t = x^{t_0}). \tag{1.6}$$

$$f(x) = \begin{cases} f_1(x_1,x_2,x_3,x_4,x_5) &= x_1.\overline{x_2} + \overline{x_1}.x_2 \\ f_2(x_1,x_2,x_3,x_4,x_5) &= \overline{x_1 + x_2} \\ f_3(x_1,x_2,x_3,x_4,x_5) &= x_3.\overline{x_1} \\ f_4(x_1,x_2,x_3,x_4,x_5) &= x_5 \\ f_5(x_1,x_2,x_3,x_4,x_5) &= \overline{x_3} + x_4 \end{cases}$$

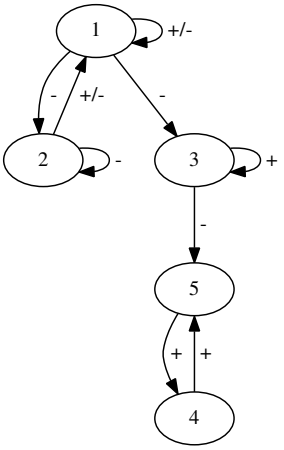


FIGURE 1.3 – Définition de $f : \mathbb{B}^5 \to \mathbb{B}^5$ et son graphe d'interaction

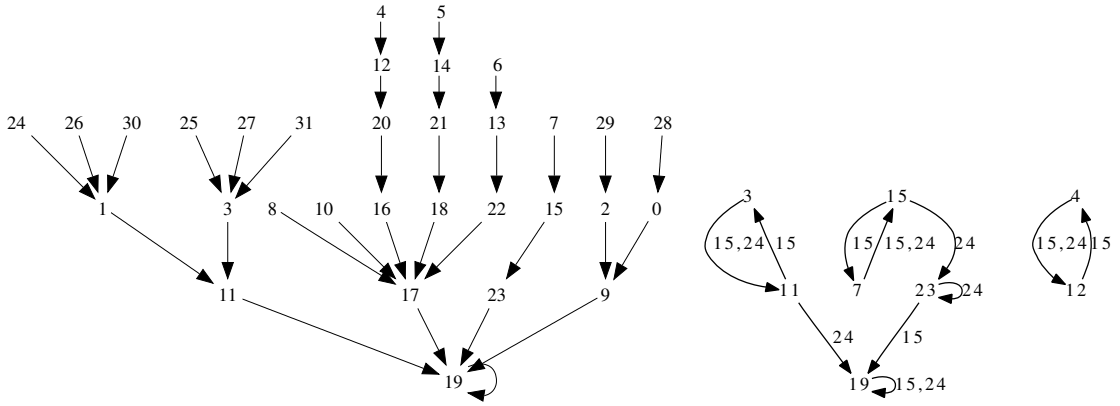
Sinon les itérations sont dites *divergentes*. De plus, si $(x^{(t)})^{t\in\mathbb{N}}$ défini selon l'équation (1.5) satisfait (1.6) pour tous les $x^{(0)} \in E$, pour toutes les stratégies pseudo-périodiques s et pour toutes les matrices de dates, $(D^{(t)})^{t\in\mathbb{N}}$, alors les itérations de f sont *universellement convergentes*.

Exemple. *On considère cinq éléments à valeurs dans* $\mathbb{B}$*. Une configuration dans* $\mathbb{B}^5$ *est représentée par un entier entre 0 et 31. La* FIGURE *1.3 donne la fonction définissant la dynamique du système et son graphe d'interaction. On note que le graphe d'interaction contient cinq cycles. Les résultats connus [Bah00] de conditions suffisantes établissant la convergence du système pour les itérations généralisées sont basés sur l'absence de cycles. Ils ne peuvent donc pas être appliqués ici.*

Dans ce qui suit, les configurations sont représentées à l'aide d'entiers plutôt que nombres binaires. Le graphe des itérations synchrones est donné en FIGURE 1.4(a). Depuis n'importe quelle configuration, on constate qu'il converge vers le point fixe correspondant à l'entier 19. Un extrait du graphe des itérations unaires est donné à la FIGURE 1.4(b). Les libellés des arcs correspondent aux éléments activés. Les itérations unaires ne convergent pas pour la stratégie pseudo-périodique donnée à l'équation (1.4) : le système peut infiniment boucler entre 11 et 3, entre 15 et 7.

Comme les itérations unaires ne convergent pas pour certaines stratégies, les itérations asynchrones basées sur les même stratégies peuvent ne pas converger non plus. Cependant, même si l'on considère que tous les composants sont activés à chaque itération, c'est à dire si s^t est constamment égal à $2^n - 1$, le délai peut introduire de la divergence. On considère par exemple la matrice D^t dont chaque élément vaut t sauf D^t_{12} qui vaut $t-1$ si t est impair. On a ainsi $x^{t+1} = f(x^t)$ si t est pair et

$$x^{t+1} = \left(f_1(x_1^t, x_2^{t-1}, x_3^t, x_4^t, x_5^t), f_2(x^t), \ldots, f_5(x^t)\right).$$



(a) Itérations synchrones de f. (b) Extrait des itérations unaires.

FIGURE 1.4 – Graphes des itérations de f définie à la figure 1.3

sinon. En démarrant de $x^0 = 00011$, le système atteint $x^1 = 01011$ et boucle entre ces deux configurations. Pour une même stratégie, les itérations asynchrones divergent alors que les synchrones convergent. Les sections suivantes de ce chapitre montrent comment résoudre ce problème.

1.2.1/ ITÉRATIONS MIXTES

Introduit dans [ABCVS05] le mode d'*itérations mixtes* combine synchronisme et asynchronisme. Intuitivement, les nœuds qui pourraient introduire des cycles dans les itérations asynchrones sont regroupés. Les noeuds à l'intérieur de chaque groupe seront itérés de manière synchrone. Les itérations asynchrones sont conservées entre les groupes.

Définition 1 (Relation de Synchronisation)**.** *Soit une fonction f et $\Gamma(f)$ son graphe d'interaction. La* relation de synchronisation *$\mathcal{R}$ est définie sur l'ensemble des nœuds par : $i\mathcal{R}j$ si i et j appartiennent à la même composante fortement connexe (CFC) dans $\Gamma(f)$.*

On peut facilement démontrer que la relation de synchronisation est une relation d'équivalence sur l'ensemble des éléments. On introduit quelques notations : par la suite $\langle i \rangle$ représente la classe d'équivalence de i et $\mathcal{K}$ représente l'ensemble de toutes les classes, *i.e.*, $\mathcal{K} = \{1, \ldots, n\}/\mathcal{R}$. On peut définir les itérations mixtes.

Définition 2 (Itérations mixtes)**.** *Les itérations mixtes d'un système discret suivent l'équation (1.5) où de plus $bin(s^t)[i] = bin(s^t)[j]$ et $D^t_{ij} = D^t_{ji} = t$ si $i\mathcal{R}j$.*

Dans ce contexte, il n'y a plus de délai entre deux noeuds de la même CFC et leurs mises à jour sont synchronisées. Cependant, pour p_0 et p_1 dans la même classe $\langle p \rangle$, et q dans une autre classe $\langle q \rangle$, ce mode opératoire autorise des délais différents entre p_0 et q et entre p_1 et q. Ainsi p_1 et p_2 sont distinguables même s'ils appartiennent à la même classe. Pour gommer cette distinction, on définit le mode suivant :

Définition 3 (Itérations mixtes avec delais uniformes)**.** *Le mode mixte a des* délais uniformes *si pour chaque $t = 0, 1, \ldots$ et pour chaque paire de classes $(\langle p \rangle, \langle q \rangle)$, il existe une*

constante d^t_{pq} telle que la propriété suivante est établie :

$$\bigwedge_{p_k \in \langle p \rangle, q_k \in \langle q \rangle} D^t_{p_k q_k} = d^t_{pq}$$

On a alors le théorème suivant.

Théorème [5]. *Soit une fonction f possédant un unique point fixe x^* et une stratégie pseudo-périodique s. Si les itérations synchrones convergent vers x^* pour cette stratégie, alors les itérations mixtes à délai uniforme convergent aussi vers x^* pour cette stratégie.*

La preuve de ce théorème est donnée en section A.1.

1.2.2/ DURÉES DE CONVERGENCE

Cette section donne des bornes supérieures et inférieures des durées globales de convergence pour les modes synchrones, mixtes et asynchrones. Pour simplifier le discours, on considère que les itérations convergent en I étapes dans le mode synchrone et que le graphe d'interaction ne contient qu'une seule composante connexe. Les durées de convergence prennent en compte les temps de calcul et les temps de communication, ce depuis l'initialisation et jusqu'à la stabilisation.

Pour simplifier l'évaluation, nous considérons que le temps de calcul d'une itération sur un composant ainsi que celui de communication entre deux composants est constant. Ceci implique en particulier que, dans le mode asynchrone, ces derniers sont bornés. En d'autres mots, il existe un entier δ_0 tel que $0 \leq t - D^t_{ij} \leq \delta_0$ est établi pour tout couple de nœuds (i, j). Les notations utilisées sont les suivantes :

- **Taille pour coder l'information :** elle représente le nombre de bits nécessaires pour représenter l'état courant du composant i et est notée cs_i ;
- **Temps de calcul :** le composant i a besoin de cp_i unités de temps pour faire une mise à jour locale de son état ;
- **Temps de communication :** on utilise le modèle classique de communication $\beta + L\tau$ où L est le nombre de bits transférés. On définit β_{ij} et τ_{ij} comme la latence et la bande passante du lien entre i et j.

1.2.3/ LE MODE SYNCHRONE

Dans le cas synchrone, la convergence la plus rapide est obtenue lorsque le point fixe x^* est accessible en un seul pas depuis toute configuration. Le temps global de convergence est donc minoré par $T_{min}(Sync) = \max_i cp_i$ Dans le cas général, si B est la matrice d'adjacence représentant le graphe d'interaction, le temps global de convergence est

$$T(Sync) = I \times (\max_i cp_i + \max_{i,j}(B_{ji} \times (\beta_{ij} + cs_i \times \tau_{ij}))) \tag{1.7}$$

Exemple. *Intuitivement la convergence se propage selon les dépendances internes au système : un nœud se stabilise lorsque ceux dont il dépend sont eux aussi stables. Cette stabilisation progressive est illustrée à la* FIGURE *1.5 qui représente des exécutions synchrones dans le cas d'une initialisation avec la valeur (00100). Dans cette figure et les suivantes, les blocs doublement hachurés indiquent la stabilisation du composant.*

On peut constater que la première classe $\langle 1 \rangle$ *se stabilise en deux itérations, la seconde classe* $\langle 3 \rangle$ *atteint sa valeur finale l'itération suivante tandis que la dernière classe,* $\langle 4 \rangle$*, converge en deux itérations.*

$$I = I_{\langle 1 \rangle} + I_{\langle 3 \rangle} + I_{\langle 4 \rangle} = 2 + 1 + 2 = 5 \tag{1.8}$$

1.2.4/ LE MODE MIXTE

On considère $|\mathcal{K}|$ classes de composants synchronisés. (comme donné en équation (1.8)). Soit I_k le nombre d'itérations suffisant pour que la classe $\langle k \rangle \in \mathcal{K}$ se stabilise sachant que toutes ses dépendances ont déjà convergé. Ainsi I vaut $\sum_{\langle k \rangle \in \mathcal{K}} I_k$. La borne inférieure pour la durée de convergence des itérations asynchrones est

$$T(\textit{Mixed}) \geq \sum_{k \in \mathcal{K}} I_k (\max_{l \in k} \textit{cp}_l) \tag{1.9}$$

qui apparaît lorsque tous les délais de communication sont consommés par des durées de calcul.

Concernant le majorant, celui-ci correspond au cas où les durées de communications entre les classes désynchronisées ne sont pas consommées par des calculs ou lorsque chaque classe nécessite la stabilisation de tous ses ascendants pour converger. On a dans ce cas :

$$T(\textit{Mixed}) \leq \sum_{k \in \mathcal{K}} \left(I_k \times (\max_{l \in k} \textit{cp}_l) + \max_{l \in k, e \in k', k \leq k'} B_{el} \times (\beta_{le} + \textit{cs}_l \tau_{le}) \right) \tag{1.10}$$

Exemple. *Une exécution du mode mixte est donnée à la* FIGURE *1.6. On peut constater que le temps d'exécution peut être plus petit que pour le mode synchrone.*

1.2.5/ LE MODE UNAIRE ASYNCHRONE

En termes de durée de convergence, ce mode peut être vu comme un cas particulier du mode mixte où toutes les classes sont des singletons. La borne minimale peut donc s'exprimer comme :

$$T(\textit{Async}) \geq \max_{i=1}^{n} I_i \times \textit{cp}_i \tag{1.11}$$

où I_i est le nombre d'itérations suffisant pour que le nœud i converge et qui est atteint si tous les nœuds sont indépendants les uns des autres. Cette borne est arbitrairement faible et n'est pas atteinte dès qu'une dépendance existe. La borne supérieure quant à elle est donnée par :

$$T(\textit{Async}) \leq \sum_{i=1}^{n} \left(I_i \times \textit{cp}_i + \max_{1 \leq k \leq n} B_{ki} (\beta_{ik} + \textit{cs}_i \tau_{ik}) \right) \tag{1.12}$$

et apparaît lorsque chaque élément dépend des autres et que les calculs ne recouvrent nullement les communications.

Exemple. *La* FIGURE *1.7 présente un exemple d'exécution du mode unaire asynchrone. Certaines communications issues de l'élément* 4 *n'ont pas été représentées pour des raisons de clarté. On constate que le temps global de convergence est plus petit que celui des deux autres modes.*

1.3/ CONCLUSION

Introduire de l'asynchronisme peut permettre de réduire le temps d'exécution global, mais peut aussi introduire de la divergence. Dans ce chapitre, après avoir introduit les bases sur les réseaux booléens, nous avons exposé comment construire un mode combinant les avantages du synchronisme en termes de convergence avec les avantages de l'asynchronisme en termes de vitesse de convergence.

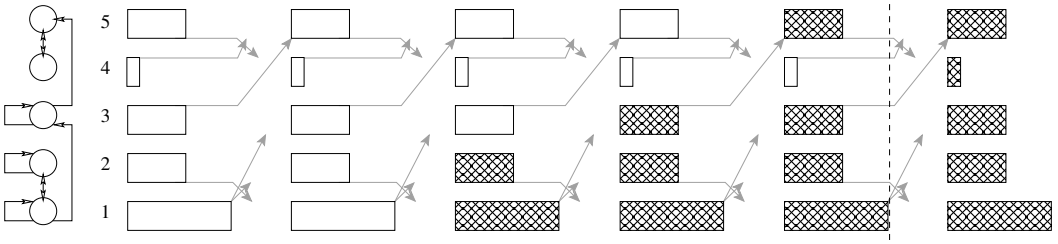


FIGURE 1.5 – Itérations synchrones

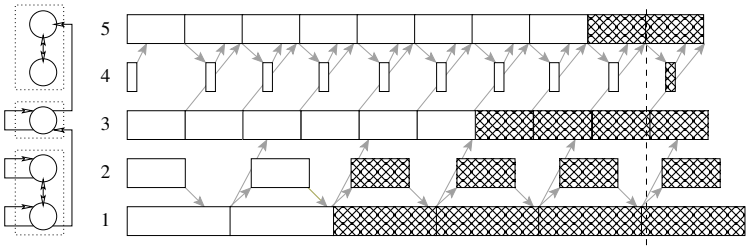


FIGURE 1.6 – Itérations mixtes avec $\langle 1\rangle = \{1,2\}$, $\langle 3\rangle = \{3\}$, $\langle 4\rangle = \{4,5\}$.

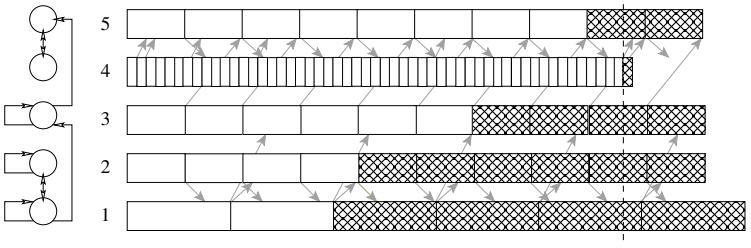


FIGURE 1.7 – Itérations asynchrones

2

PREUVE AUTOMATIQUE DE CONVERGENCE

Sur des petits exemples, l'étude de convergence de systèmes dynamiques discrets est simple à vérifier pratiquement pour le mode synchrone. Lorsqu'on introduit des stratégies pseudo-périodiques pour les modes unaires et généralisés, le problème se complexifie. C'est pire encore lorsqu'on traite des itérations asynchrones et mixtes prenant de plus en compte les délais.

Des méthodes de simulation basées sur des stratégies et des délais générés aléatoirement ont déjà été présentées [BM99, BCV02]. Cependant, comme ces implantations ne sont pas exhaustives, elles ne donnent un résultat formel que lorsqu'elles fournissent un contre-exemple. Lorsqu'elles exhibent une convergence, cela ne permet que de donner une intuition de convergence, pas une preuve. Autant que nous sachions, aucune démarche de preuve formelle automatique de convergence n'a jamais été établie. Dans le travail théorique [Cha06], Chandrasekaran a montré que les itérations asynchrones sont convergentes si et seulement si on peut construire une fonction de Lyaponov décroissante, mais il ne donne pas de méthode automatique pour construire cette fonction.

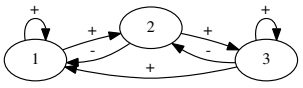
Un outil qui construirait automatiquement toutes les transitions serait le bienvenu. Pour peu qu'on établisse la preuve de correction et de complétude de la démarche, la convergence du réseau discret ne reposerait alors que sur le verdict donné par l'outil. Cependant, même pour des réseaux discrets à peu d'éléments, le nombre de configurations induites explose rapidement. Les *Model-Checkers* [Hol03, CCG+02, BHJM07, FS07, RDH03] sont des classes d'outils qui adressent le problème de détecter automatiquement si un modèle vérifie une propriété donnée. Pour traiter le problème d'explosion combinatoire, ces outils appliquent des méthodes d'ordre partiel, d'abstraction, de quotientage selon une relation d'équivalence.

Ce chapitre montre comment nous simulons des réseaux discrets pour établir formellement leur convergence (ou pas). Nous débutons par un exemple et faisons quelques rappels sur le langage PROMELA qui est le langage du model-checker SPIN [Hol03] (Section 2.1). Nous présentons ensuite la démarche de traduction de réseaux discrets dans PROMELA (Section 2.2). Les théorèmes de correction et de complétude de la démarche sont ensuite donnés à la Section 2.3. Des données pratiques comme la complexité et des synthèses d'expérimentations sont ensuite fournies (Section 2.4). Ce chapitre a fait l'objet du rapport [Cou10].

Exemple. *On considère un exemple à trois éléments dans* $\mathbb{B}$*. Chaque configuration est*

$$F(x) = \begin{cases} f_1(x_1, x_2, x_3) &= x_1.\overline{x_2} + x_3 \\ f_2(x_1, x_2, x_3) &= x_1 + \overline{x_3} \\ f_3(x_1, x_2, x_3) &= x_2.x_3 \end{cases}$$

(a) Fonction à itérer



(b) Graphe d'intéraction

FIGURE 2.1 – Exemple pour SDD ≈ SPIN.

ainsi un élément de $\mathbb{B}^3$, i.e., un nombre entre 0 et 7. La FIGURE *2.1(a) précise la fonction f considérée et la* FIGURE *2.1(b) donne son graphe d'intéraction.*

On peut facilement vérifier que toutes les itérations parallèles synchrones initialisées avec $x^0 \neq 7$ soit (111) convergent vers 2 soit (010) ; celles initialisées avec $x^0 = 7$ restent en 7. Pour les modes unaires ou généralisés avec une stratégie pseudo-périodique, on a des comportements qui dépendent de la configuration initiale :

- *initialisées avec 7, les itérations restent en 7 ;*
- *initialisées avec 0, 2, 4 ou 6 les itérations convergent vers 2 ;*
- *initialisées avec 1, 3 ou 5, les itérations convergent vers un des deux points fixes 2 ou 7.*

2.1/ RAPPELS SUR LE LANGAGE PROMELA

Cette section rappelle les éléments fondamentaux du langage PROMELA (Process Meta Language). On peut trouver davantage de détails dans [Hol03, Wei97].

```
#define N 3
#define d_0 5

bool X [N]; bool Xp [N]; int mods [N];
typedef vals{bool v [N]};
vals Xd [N];

typedef a_send{chan sent[N]=[d_0] of {bool}};
a_send channels [N];

chan unlock_elements_update=[1] of {bool};
chan sync_mutex=[1] of {bool};
```

FIGURE 2.2 – Declaration des types de la traduction.

Comme en C, on peut déclarer des tableaux à une dimension ou des nouveaux types de données (introduites par le mot clef `typedef`).

Exemple. *Le programme donné à la* FIGURE *2.2 correspond à des déclarations de variables qui servent dans l'exemple de ce chapitre. Il définit :*

- *les constantes* `N` *et* `d_0` *qui précisent respectivement le nombre* N *d'éléments et le délai maximum δ_0 ;*
- *les deux tableaux (*`X` *et* `Xp`*) de* `N` *variables booléennes ; les cellules* `X[i]` *et* `Xp[i]` *sont associées à la variables x_{i+1} d'un système dynamique discret ; elles mémorisent les valeurs de X_{i+1} respectivement avant et après sa mise à jour ; il suffit ainsi de comparer* `X` *et* `Xp` *pour constater si x a changé ou pas ;*
- *le tableau* `mods` *contient les éléments qui doivent être modifiés lors de l'itération en cours ; cela correspond naturellement à l'ensemble des éléments s^t ;*

— *le type de données structurées* `vals` *et le tableau de tableaux* `Xd[`i`].v[`j`]` *qui vise à mémoriser* $x_{j+1}^{D_{i+1j+1}^{t-1}}$ *pour l'itération au temps* t.

Déclarée avec le mot clef `chan`*, une donnée de type* `channel` *permet le transfert de messages entre processus dans un ordre FIFO. Dans l'exemple précédent, on déclare successivement :*

— *un canal* `sent` *qui vise à mémoriser* `d_0` *messages de type* `bool` *; le tableau nommé* `channels` *de* `N*N` *éléments de type* `a_send` *est utilisé pour mémoriser les valeurs intermédiaires* x_j *; Il permet donc de temporiser leur emploi par d'autres éléments* i.

— *les deux canaux* `unlock_elements_update` *et* `sync_mutex` *contenant chacun un message booléen et sont utilisés ensuite comme des sémaphores.*

Le langage PROMELA exploite la notion de *process* pour modéliser la concurrence au sein de systèmes. Un process est instancié soit immédiatement (lorsque sa déclaration est préfixée par le mot-clef `active`) ou bien au moment de l'exécution de l'instruction `run`. Parmi tous les process, `init` est le process initial qui permet d'initialiser les variables, lancer d'autres process. . .

Les instructions d'affectation sont interprétées usuellement. Les canaux sont concernés par des instructions particulières d'envoi et de réception de messages. Pour un canal `ch`, ces instructions sont respectivement notées `ch ! m` et `ch ? m`. L'instruction de réception consomme la valeur en tête du canal `ch` et l'affecte à la variable `m` (pour peu que `ch` soit initialisé et non vide). De manière similaire, l'instruction d'envoi ajoute la valeur de `m` à la queue du canal `ch` (pour peu que celui-ci soit initialisé et pas plein). Dans les cas problématiques, canal non initialisé et vide pour une réception ou plein pour un envoi, le processus est bloqué jusqu'à ce que les conditions soient remplies.

La structures de contrôle `if` (resp. `do`) définit un choix non déterministe (resp. une boucle non déterministe). Que ce soit pour la conditionnelle ou la boucle, si plus d'une des conditions est établie, l'ensemble des instructions correspondantes sera choisi aléatoirement puis exécuté.

Dans le process `init` détaillé à la FIGURE 2.3, une boucle de taille `N` initialise aléatoirement la variable globale de type tableau `Xp`. Ceci permet par la suite de vérifier si les itérations sont convergentes pour n'importe quelle configuration initiale x^0.

Pour chaque élément i, si les itérations sont asynchrones

— on stocke d'abord la valeur de `Xp[i]` dans chaque `Xd[j].v[i]` puisque la matrice s^0 est égale à (0),

— puis, la valeur de i (représentée par `Xp[i]`) devrait être transmise à j s'il y a un arc de i à j dans le graphe d'incidence. Dans ce cas, c'est la fonction `hasnext` (détaillée à la FIGURE 2.5) qui mémorise ce graphe en fixant à `true` la variable `is_succ`, naturellement et à `false` dans le cas contraire. Cela permet d'envoyer la valeur de i dans le canal au travers de `channels[i].sent[j]`.

2.2/ DU SYSTÈME BOOLÉEN AU MODÈLE PROMELA

Les éléments principaux des itérations asynchrones rappelées à l'équation (1.5) sont la stratégie, la fonctions et la gestion des délais. Dans cette section, nous présentons successivement comment chacune de ces notions est traduite vers un modèle PROMELA.

```
init{
 int i=0; int j=0; bool is_succ=0;
 do
 ::i==N->break;
 ::i< N->{
   if
   ::Xp[i]=0;
   ::Xp[i]=1;
   fi;
   j=0;
   do
   ::j==N -> break;
   ::j< N -> Xd[j].v[i]=Xp[i]; j++;
   od;
   j=0;
   do
   ::j==N -> break;
   ::j< N -> {
     hasnext(i,j);
     if
     ::(i!=j && is_succ==1) ->
     channels[i].sent[j] ! Xp[i];
     ::(i==j || is_succ==0) -> skip;
     fi;
     j++;}
   od;
   i++;}
 od;
 sync_mutex ! 1;
}
```

FIGURE 2.3 – Process init.

```
active proctype scheduler(){
 do
 ::sync_mutex ? 1 -> {
   int i=0; int j=0;
   do
   :: i==N -> break;
   :: i< N -> {
     if
     ::skip;
     ::mods[j]=i; j++;
     fi;
     i++;}
   od;
   ar_len=i;
   unlock_elements_update ! 1;
   }
 od
}
```

FIGURE 2.4 – Process scheduler pour la stratégie pseudo-périodique.

```
inline hasnext(i,j){
  if
    :: i==0 && j ==0 -> is_succ = 1
    :: i==0 && j ==1 -> is_succ = 1
    :: i==0 && j ==2 -> is_succ = 0
    :: i==1 && j ==0 -> is_succ = 1
    :: i==1 && j ==1 -> is_succ = 0
    :: i==1 && j ==2 -> is_succ = 1
    :: i==2 && j ==0 -> is_succ = 1
    :: i==2 && j ==1 -> is_succ = 1
    :: i==2 && j ==2 -> is_succ = 1
  fi
}
```

FIGURE 2.5 – Codage du graphe d'intéraction de f.

2.2.1/ LA STRATÉGIE

Regardons comment une stratégie pseudo-périodique peut être représentée en PROMELA. Intuitivement, un process `scheduler` (comme représenté à la FIGURE 2.4) est itérativement appelé pour construire chaque s^t représentant les éléments possiblement mis à jour à l'itération t.

Basiquement, le process est une boucle qui est débloquée lorsque la valeur du sémaphore `sync_mutex` est 1. Dans ce cas, les éléments à modifier sont choisis aléatoirement (grâce à N choix successifs) et sont mémorisés dans le tableau `mods`, dont la taille est `ar_len`. Dans la séquence d'exécution, le choix d'un élément mis à jour est directement suivi par des mises à jour : ceci est réalisé grâce à la modification de la valeur du sémaphore `unlock_elements_updates`.

2.2.2/ ITÉRER LA FONCTION f

La mise à jour de l'ensemble $s^t = \{s_1, \ldots, s_m\}$ des éléments qui constituent la stratégie $(s^t)^{t\in\mathbb{N}}$ est implantée à l'aide du process `update_elems` fourni à la FIGURE 2.7. Ce processus actif attend jusqu'à ce qu'il soit débloqué par le process `scheduler` à l'aide du sémaphore `unlock_elements_update`. L'implantation se déroule en cinq étapes :

1. elle commence en mettant à jour la variable `X` avec les valeurs de `Xp` dans la fonction `update_X`, FIGURE 2.6
2. elle mémorise dans `Xd` la valeur disponible pour chaque élément grâce à la fonction `fetch_values` ; cette fonction est détaillée dans la section suivante ;
3. une boucle met à jour itérativement la valeur de j (grâce à l'appel de fonction `f(j)`) pour peu que celui-ci doive être modifié, *i.e.*, pour peu qu'il soit renseigné dans `mods[count]` ; le code source de `F` est donné en FIGURE 2.8 et est une traduction directe de l'application f ;

```
inline update_X(){
  int countu;
  countu = 0;
  do
    :: countu == N -> break ;
    :: countu != N ->
       X[countu] = Xp[countu];
       countu ++ ;
  od
}
```

FIGURE 2.6 – Sauvegarde de l'état courant

```
active proctype update_elems(){
 do
 ::unlock_elements_update ? 1 ->
   {
    atomic{
     bool is_succ=0;
     update_X();
     fetch_values();
     int count = 0;
     int j = 0;
     do
     ::count == ar_len -> break;
     ::count < ar_len ->
       j = mods[count];
       F(j);
       count++;
     od;
     diffuse_values(Xp);
     sync_mutex ! 1
    }
   }
 od
}
```

FIGURE 2.7 – Mise à jour des éléments

```
inline F(){
 if
 ::j==0 -> Xp[0] =
      (Xs[j].v[0] & !Xs[j].v[1])
     |(Xs[j].v[2])
 ::j==1 -> Xp[1] =  Xs[j].v[0]
                  | !Xs[j].v[2]
 ::j==2 -> Xp[2] =  Xs[j].v[1]
                  &  Xs[j].v[2]
 fi
}
```

FIGURE 2.8 – Application de la fonction f

4. les nouvelles valeurs des éléments `Xp` sont symboliquement envoyés aux autres éléments qui en dépendent grâce à la fonction `diffuse_values(Xp)` ; cette dernière fonction est aussi détaillée dans la section suivante ;
5. finalement, le process informe le scheduler de la fin de la tâche (au travers du sémaphore `sync_mutex`).

2.2.3/ GESTION DES DÉLAIS

Cette section montre comment les délais inhérents au mode asynchrone sont traduits dans le modèle PROMELA grâce à deux fonctions `fetch_values` et `diffuse_values`. Celles-ci sont données en FIGURE 2.9 et 2.10. Elles récupèrent et diffusent respectivement les valeurs des éléments.

La première fonction met à jour le tableau `Xd` requis pour les éléments qui doivent être modifiés. Pour chaque élément dans `mods`, identifié par la variable j, la fonction récupère les valeurs des autres éléments (dont le libellé est i) dont j dépend. Il y a deux cas.

- puisque i connaît sa dernière valeur (*i.e.*, D^t_{ii} est toujours t) `Xd[i].v[i]` est donc `Xp[i]` ;
- sinon, il y a deux sous-cas qui peuvent peuvent potentiellement modifier la valeur que j a de i (et qui peuvent être choisies de manière aléatoire) :
  - depuis la perspective de j la valeur de i peut ne pas avoir changé (c'est l'instruction `skip`) ou n'est pas utile ; ce dernier cas apparaît lorsqu'il n'y a pas d'arc de i à j dans le graphe d'incidence, *i.e.*, lorsque la valeur de `is_succ` qui est calculée par `hasnext(i,j)` est 0 ; dans ce cas, la valeur de `Xd[j].v[i]` n'est pas modifiée ;
  - sinon, on affecte à `Xd[j].v[i]` la valeur mémorisée dans le canal `channels[i].sent[j]` (pour peu que celui-ci ne soit pas vide).

Les valeurs des éléments sont ajoutées dans ce canal au travers de la fonction `diffuse_values`. L'objectif de cette fonction est de stocker les valeurs de x (représenté dans le modèle par `Xp`) dans le canal `channels`. Il permet au model-checker SPIN d'exécuter le modèle PROMELA comme s'il pouvait y avoir des délais entre processus. Il y a deux cas différents pour la valeur de X_j :

```
inline fetch_values(){
 int countv = 0;
 do
 :: countv == ar_len -> break ;
 :: countv < ar_len ->
    j = mods[countv];
    i = 0;
    do
    :: (i == N) -> break;
    :: (i < N && i == j) -> {
        Xd[j].v[i] = Xp[i] ;
        i++ }
    :: (i < N && i != j) -> {
        hasnext(i,j);
        if
        :: skip
        :: is_succ==1 &&
    nempty(channels[i].sent[j]) ->
             channels[i].sent[j] ?
                        Xd[j].v[i];
        fi;
        i++ }
    od;
    countv++
 od
}
```

FIGURE 2.9 – Récupérer les valeurs des elements

```
inline diffuse_values(values){
 int countb=0;
 do
 :: countb == ar_len -> break ;
 :: countb < ar_len ->
    j = mods[countb];
    i = 0 ;
    do
    :: (i == N) -> break;
    :: (i < N && i == j) -> i++;
    :: (i < N && i != j) -> {
        hasnext(j,i);
        if
        :: skip
        :: is_succ==1 &&
     nfull(channels[j].sent[i]) ->
            channels[j].sent[i] !
                        values[j];
        fi;
        i++ }
    od;
    countb++
 od
}
```

FIGURE 2.10 – Diffuser les valeurs des elements

— soit elle est « perdue », « oubliée » pour permettre à i de ne pas tenir compte d'une des valeurs de j ; ce cas a lieu soit lors de l'instruction `skip` ou lorsqu'il n'y a pas d'arc de j à i dans le graphe d'incidence ;
— soit elle est mémorisée dans le canal `channels[j].sent[i]` (pour peu que celui-ci ne soit pas plein).

L'introduction de l'indéterminisme à la fois dans les fonctions `fetch_values` et `diffuse_values` est nécessaire dans notre contexte. Si celui-ci n'était présent que dans la fonction `fetch_values`, nous ne pourrions pas par exemple récupérer la valeur $x_i^{(t)}$ sans considérer la valeur $x_i^{(t-1)}$. De manière duale, si le non déterminisme était uniquement utilisé dans la fonction `diffuse_values`, alors chaque fois qu'une valeur serait mise dans le canal, elle serait immédiatement consommée, ce qui est contradictoire avec la notion de délai.

2.2.4/ PROPRIÉTÉ DE CONVERGENCE UNIVERSELLE

Il reste à formaliser dans le model checker SPIN le fait que les itérations d'un système dynamique à N éléments est universellement convergent.

Rappelons tout d'abord que les variables `X` et `Xp` contiennent respectivement la valeur de x avant et après la mise à jour. Ainsi, si l'on effectue une initialisation non déterministe de `Xp` et si l'on applique une stratégie pseudo-périodique, il est nécessaire et suffisant de prouver la formule temporelle linéaire (LTL) suivante :

$$\diamond\,(\square \mathtt{Xp} = \mathtt{X}) \tag{2.1}$$

où les opérateurs $\diamond$ et $\square$ ont la sémantique usuelle, à savoir respectivement *éventuellement* et *toujours* dans les chemins suivants. On note que cette propriété, si elle est établie, garantit la stabilisation du système. Cependant elle ne donne aucune métrique quant à la manière dont celle-ci est obtenue. En particulier, on peut converger très lentement ou le système peut même disposer de plusieurs points fixes.

2.3/ CORRECTION ET COMPLÉTUDE DE LA DÉMARCHE

Cette section présente les théorèmes de correction et de complétude de l'approche (Théorèmes 6 et 7). Toutes les preuves sont déplacées en annexe A.2.

Théorème 6 (Correction de la traduction vers Promela)**.** *Soit ϕ un modèle de système dynamique discret et ψ sa traduction PROMELA. Si ψ vérifie la propriété LTL (2.1) sous hypothèse d'équité faible, alors les itérations de ϕ sont universellement convergentes.*

Théorème 7 (Complétude de la traduction vers Promela)**.** *Soit ϕ un modèle de système dynamique discret et ψ sa traduction. Si ψ ne vérifie pas la propriété LTL (2.1) sous hypothèse d'équité faible, alors les itérations de ϕ ne sont pas universellement convergentes.*

2.4/ DONNÉES PRATIQUES

Cette section donne tout d'abord quelques mesures de complexité de l'approche puis présente ensuite les expérimentations issues de ce travail.

Théorème 8 (Nombre d'états)**.** *Soit ϕ un modèle de système dynamique discret à* N *éléments, m arcs dans le graphe d'incidence et ψ sa traduction en PROMELA. Le nombre de configurations de l'exécution en SPIN de ψ est borné par $2^{m(\delta_0+1)+n(n+2)}$.*

Démonstration. Une configuration est une évaluation des variables globales. Leur nombre ne dépend que de celles qui ne sont pas constantes.

Les variables `Xp` et `X` engendrent 2^{2n} états. La variable `Xs` génère 2^{n^2} états. Chaque canal de `array_of_channels` peut engendrer $1 + 2^1 + \ldots + 2^{\delta_0} = 2^{\delta_0+1} - 1$ états. Puisque le nombre d'arêtes du graphe d'incidence est m, il y a m canaux non constants, ce qui génère approximativement $2^{m(\delta_0+1)}$ états. Le nombre de configurations est donc borné par $2^{m(\delta_0+1)+n(n+2)}$. On remarque que cette borne est traitable par SPIN pour des valeurs raisonnables de N, m et δ_0.

□

La méthode détaillée ici a pu être appliquée sur l'exemple pour prouver formellement sa convergence universelle.

On peut remarquer que SPIN n'impose l'équité faible qu'entre les process alors que les preuves des deux théorèmes précédents reposent sur le fait que cette équité est établie dès qu'un choix indéterministe est effectué. Naïvement, on pourrait considérer comme hypothèse la formule suivante chaque fois qu'un choix indéterministe se produit entre k événements respectivement notés $l_1, \ldots l_k$:

$$\Box \diamond (l == l_0) \Rightarrow ((\Box \diamond (l == l_1)) \wedge \ldots \wedge (\Box \diamond (l == l_k)))$$

où le libellé l_0 dénote le libellé de la ligne précédent le choix indéterministe. Cette formule traduit exactement l'équité faible. Cependant en raison de l'explosion de la taille du produit entre l'automate de Büchi issu de cette formule et celui issu du programme

PROMELA, SPIN n'arrive pas à vérifier si la convergence universelle est établie ou non sur des exemples simples.

Ce problème a été pratiquement résolu en laissant SPIN générer toutes les traces d'exécution, même celles qui ne sont pas équitables, puis ensuite vérifier la propriété de convergence sur toutes celles-ci. Il reste alors à interpréter les résultats qui peuvent être de deux types. Si la convergence est établie pour toutes les traces, elle le reste en particulier pour les traces équitables. Dans le cas contraire on doit analyser le contre exemple produit par SPIN.

La méthode détaillée ici a été appliquée sur des exemples pour prouver formellement leur convergence ou leur divergence (FIGURE 2.11) avec ou sans délais. Dans ces expériences, les délais ont été bornés par $\delta_0 = 10$. Dans ce tableau, P est vrai (1.8) si et seulement si la convergence universelle est établie et faux $(\perp)$ sinon. Le nombre M est la taille de la mémoire consommée (en MB) et T est le temps d'exécution sur un Intel Centrino Dual Core 2 Duo @1.8GHz avec 2GB de mémoire vive pour établir un verdict.

	Synchrones			Généralisées		
	P	M	T	P	M	T
RE	1.8	2.7	0.01s	$\perp$	369.371	0.509s
[RC07]	$\perp$	2.5	0.001s	$\perp$	2.5	0.01s
[BM99]	1.8	36.7	12s	1.8		

(a) Sans délais

	Mode Mixte						Seulement borné					
	Synchrones			Pseudo-Périodique			Synchrones			Pseudo-Périodique		
	P	M	T	P	M	T	P	M	T	P	M	T
RE	1.8	409	1m11s	$\perp$	370	0.54	$\perp$	374	7.7s	$\perp$	370	0.51s
[RC07]	$\perp$	2.5	0.001s	$\perp$	2.5	0.01s	$\perp$	2.5	0.01s	$\perp$	2.5	0.01s
[BM99]	1.8			1.8			$\perp$			$\perp$		

(b) Avec délais

FIGURE 2.11 – Résultats des simulations Promela des SDDs

L'exemple *RE* est l'exemple de ce chapitre, [RC07] concerne un réseau composé de deux gènes à valeur dans $\{0, 1, 2\}$ et [BM99] consiste en 10 process qui modifient leurs valeurs booléennes dans un graphe d'adjacence proche du graphe complet.

L'exemple *RE* a été prouvé comme universellement convergent. Comme la convergence n'est déjà pas établie pour les itérations synchrones de [RC07], il en est donc de même pour les itérations asynchrones. La FIGURE 2.12 donne une trace de la sortie de SPIN menant à la violation de la convergence. Celle-ci correspond à une stratégie périodique qui répète $\{1, 2\}; \{1, 2\}; \{1\}; \{1, 2\}; \{1, 2\}$ et débute avec $x = (0, 0)$. En raison de la dépendance forte entre les éléments de [BM99], δ_0 est réduit à 1. Cela aboutit cependant à 2^{100} configurations dans le mode des itérations asynchrones, montrant les limites de l'approche.

2.5/ CONCLUSION

L'idée principale de ce chapitre est que l'on peut, pour des réseaux booléens à délais bornés de petite taille, obtenir une preuve de la convergence ou de sa divergence et ce de manière automatique. L'idée principale est de traduire le réseau en PROMELA et de laisser le model checker établir la preuve. Toute l'approche a été prouvée : le verdict rendu par l'approche a donc valeur de vérité. L'approche a cependant ses limites et ne peut donc pas être apliquée qu'à des modèles simplifiés de programmes. La suite de ce

travail consiste à se focaliser sur les systèmes qui ne convergent pas.

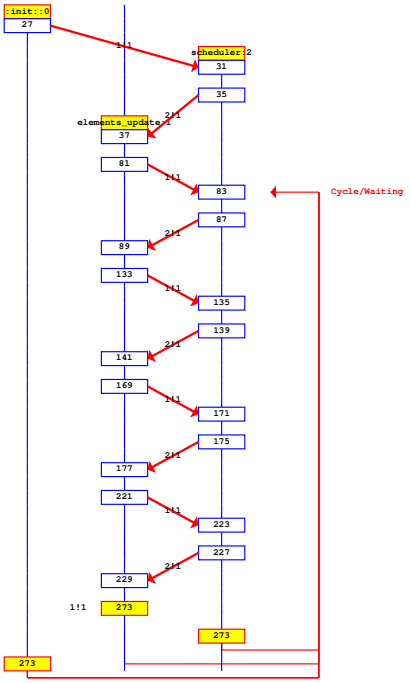


FIGURE 2.12 – Contre exemple de convergence pour [RC07]

II

DES SYSTÈMES DYNAMIQUES DISCRETS AU CHAOS

3

CARACTÉRISATION DES SYSTÈMES DISCRETS CHAOTIQUES POUR LES SCHÉMAS UNAIRES ET GÉNÉRALISÉS

La suite de ce document se focalise sur des systèmes dynamiques discrets qui ne convergent pas. Parmi ceux-ci se trouvent ceux qui sont « chaotiques ». La première section de ce chapitre rappelle ce que sont les systèmes dynamiques chaotiques et leurs caractéristiques. La section 3.2, qui est une reformulation de [Guy10], se focalise sur le schéma unaire. Elle est rappelée pour avoir un document se suffisant à lui-même. La section 3.3 étend ceci au mode généralisé. Pour chacun de ces modes, une métrique est définie. Finalement, la section 3.4 exhibe des conditions suffisantes permettant d'engendrer des fonctions chaotiques selon le mode unaire. Les sections 3.2 et 3.4 ont été publiées dans [BCG12b, BCGR11].

3.1/ SYSTÈMES DYNAMIQUES CHAOTIQUES SELON DEVANEY

Dans cette partie, les définitions fondamentales liées au chaos dans les systèmes dynamiques sont rappelées et plusieurs résultats théoriques sont montrés.

Définition 4 (Chaos (Devaney))**.** *Une fonction k continue sur un espace métrique $(\mathcal{X}, d)$ est **chaotique** si elle est transitive, régulière et fortement sensible aux conditions initiales.*

Définition 5 (Transitivité)**.** *Une fonction k est **transitive** sur $(\mathcal{X}, d)$ si la propriété suivante est établie :*

$$\forall X, Y \in \mathcal{X}, \forall \epsilon > 0, \exists Z \in \mathcal{X}, \exists t \in \mathbb{N}, d(X, Z) < \epsilon \wedge k^t(Z) = Y$$

Définition 6 (Point périodique)**.** *Un point $P \in \mathcal{X}$ est dit **périodique** de période t pour une fonction k si t est un entier naturel non nul tel que $k^t(P) = P$ et pour tout n, $0 < n \leq t - 1$, on a $k^n(P) \neq P$. Par la suite, **Per(k)** dénote l'ensemble des points périodiques de k dans $\mathcal{X}$ de période quelconque.*

Définition 7 (Régularité)**.** *Une fonction k est dite **régulière** dans $(\mathcal{X}, d)$ si l'ensemble des points périodiques de k est dense dans $\mathcal{X}$, c'est-à-dire si la propriété suivante est établie :*

$$\forall X \in \mathcal{X}, \forall \epsilon > 0, \exists Y \in \mathit{Per}(k) \textit{ tel que } d(X, Y) < \epsilon.$$

Définition [8] (Forte sensibilité aux conditions initiales)**.** *Une fonction k définie sur $(\mathcal{X}, d)$ est **fortement sensible aux conditions initiales** s'il existe une valeur $\epsilon > 0$ telle que pour tout $X \in \mathcal{X}$ et pour tout $\delta > 0$, il existe $Y \in \mathcal{X}$ et $t \in \mathbb{N}$ qui vérifient $d(X, Y) < \delta$ et $d(k^t(X), k^t(Y)) > \epsilon$.*

John Banks et ses collègues ont cependant démontré que la sensibilité aux conditions initiales est une conséquence de la régularité et de la transitivité topologique [BBCS92]. On ne se focalise donc dans la suite que sur ces deux dernières propriétés pour caractériser les fonctions booléennes f rendant chaotique la fonction engendrée G_f.

3.2/ SCHÉMA UNAIRE

Soit N un entier naturel et f une fonction de $\mathbb{B}^N$ dans lui-même.

3.2.1/ DES ITÉRATIONS UNAIRES AUX ITÉRATIONS PARALLÈLES

Dans le schéma unaire, à la $t^{\text{ème}}$ itération, seul le $s_t^{\text{ème}}$ composant (entre 1 et N) est mis à jour. Pour une stratégie $s = (s_t)_{t\in\mathbb{N}}$ (*i.e.*, une séquence d'indices de [N]), on peut définir la fonction $F_{f_u} : \mathbb{B}^N \times [N]$ vers $\mathbb{B}^N$ par

$$F_{f_u}(x, i) = (x_1, \ldots, x_{i-1}, f_i(x), x_{i+1}, \ldots, x_N). \tag{3.1}$$

Dans le schéma des itérations unaires pour une configuration initiale $x^0 \in \mathbb{B}^N$ et une stratégie $s \in [N]^{\mathbb{N}}$, les configurations x^t sont définies par la récurrence

$$x^{t+1} = F_{f_u}(x^t, s_t). \tag{3.2}$$

On peut alors construire l'espace $\mathcal{X}_u = \mathbb{B}^N \times [N]^{\mathbb{N}}$ et la fonction d'itération G_{f_u} définie de $\mathcal{X}_u$ dans lui-même par

$$G_{f_u}(x, s) = (F_{f_u}(x, s_0), \sigma(s)). \tag{3.3}$$

Dans cette définition, la fonction $\sigma : [N]^{\mathbb{N}} \longrightarrow [N]^{\mathbb{N}}$ décale la stratégie fournie en argument d'un élément vers la gauche en supprimant l'élément de tête. Ceci se formalise par

$$\sigma((u^k)_{k\in\mathbb{N}}) = (u^{k+1})_{k\in\mathbb{N}}.$$

Ainsi, effectuer des itérations unaires sur la fonction f selon une stratégie s revient à effectuer des itérations parallèles de la fonction G_{f_u} dans $\mathcal{X}_u$. La section suivante introduit une métrique sur $\mathcal{X}_u$.

3.2.2/ UNE MÉTRIQUE POUR $\mathcal{X}_u$

Sur $\mathcal{X}_u$, on définit la distance d entre les points $X = (x, s)$ et $X' = (x', s')$ de $\mathcal{X}_u$ par

$$d(X, X') = d_H(x, x') + d_S(s, s'), \text{ où } \begin{cases} d_H(x, x') = \displaystyle\sum_{i=1}^{n} |x_i - x'_i| \\ d_S(s, s') = \dfrac{9}{n} \displaystyle\sum_{t\in\mathbb{N}} \dfrac{|s_t - s'_t|}{10^{t+1}}. \end{cases} . \tag{3.4}$$

On note que dans le calcul de $d_H(x, x')$– appelée distance de Hamming entre x et x'– les termes x_i et x'_i sont considérés comme des entiers naturels égaux à 0 ou à 1 et que le calcul est effectué dans $\mathbb{Z}$. De plus, la partie entière $\lfloor d(X, X') \rfloor$ est égale à $d_H(x, x')$ soit la distance de Hamming entre x et x'. On remarque que la partie décimale est inférieure à 10^{-l} si et seulement si les l premiers termes des deux stratégies sont égaux. De plus, si la $(l+1)^{\text{ème}}$ décimale de $d_S(s, s')$ n'est pas nulle, alors s_l est différent de s'_l.

Se pose la question de caractériser les fonctions f telles que les itérations de G_{f_u} associées à leurs itérations unaires sont chaotiques dans $\mathcal{X}_u$. La section suivante apporte une réponse à cette question.

3.2.3/ CARACTÉRISATION DES FONCTIONS RENDANT CHAOTIQUES G_{f_u} SUR $\mathcal{X}_u$

Pour caractériser les fonctions rendant chaotiques dans $\mathcal{X}_u$ les itérations de G_{f_u} on se focalise donc sur la régularité et sur la transitivité de G_{f_u}. Ceci se réalise en établissant les relations d'inclusion entre les ensembles $\mathcal{T}$ des fonctions topologiquement transitives, $\mathcal{R}$ des fonctions régulières et $\mathcal{C}$ des fonctions chaotiques définis respectivement ci-dessous :

— $\mathcal{T} = \left\{ f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}} \text{ t. q. } G_{f_u} \text{ est transitive} \right\}$,
— $\mathcal{R} = \left\{ f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}} \text{ t. q. } G_{f_u} \text{ est régulière} \right\}$,
— $\mathcal{C} = \left\{ f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}} \text{ t. q. } G_{f_u} \text{ est chaotique} \right\}$.

On énonce les théorèmes successifs suivants dont les preuves sont données dans [Guy10].

Théorème [9]. *G_{f_u} est transitive si et seulement si* GIU(f) *est fortement connexe.*

Théorème [10]. $\mathcal{T} \subset \mathcal{R}$.

On peut conclure que $\mathcal{C} = \mathcal{R} \cap \mathcal{T} = \mathcal{T}$. On a alors la caractérisation suivante :

Théorème [11]. *Soit $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$. La fonction G_{f_u} est chaotique si et seulement si* GIU(f) *est fortement connexe.*

3.3/ SCHÉMA GÉNÉRALISÉ

On reprend ici le même plan que dans la section précédente.

3.3.1/ DES ITÉRATIONS GÉNÉRALISÉES AUX ITÉRATIONS PARALLÈLES

Dans le schéma généralisé, à la $t^{\text{ème}}$ itération, c'est l'ensemble des $s_t^{\text{ème}}$ éléments (inclus dans [N]) qui sont mis à jour (cf. équation (1.2)). On redéfinit la fonction $F_{f_g} : \mathbb{B}^{\mathsf{N}} \times \mathcal{P}(\{1, \ldots, \mathsf{N}\}) \to \mathbb{B}^{\mathsf{N}}$ par

$$F_{f_g}(x, s)_i = \begin{cases} f_i(x) \text{ si } i \in s ; \\ x_i \text{ sinon.} \end{cases}$$

Dans ce schéma d'itérations généralisées, pour une configuration initiale $x^0 \in \mathbb{B}^{\mathsf{N}}$ et une stratégie $S = (s_t)_{t \in \mathbb{N}} \in \mathcal{P}(\{1, \ldots, \mathsf{N}\})^{\mathbb{N}}$, les configurations x^t sont définies par la récurrence

$$x^{t+1} = F_{f_g}(s_t, x^t). \tag{3.5}$$

Soit alors G_{f_g} une fonction de $\mathbb{B}^N \times \mathcal{P}(\{1,\dots,N\})^{\mathbb{N}}$ dans lui-même définie par

$$G_{f_g}(x,S) = (F_{f_g}(x,s_0), \sigma(S)),$$

où la fonction σ est définie comme à la section précédente. A nouveau, les itérations généralisées de f induites par x^0 et la stratégie S décrivent la même orbite que les itérations parallèles de G_{f_g} depuis un point initial $X^0 = (x^0, S)$

On construit cette fois ci l'espace $\mathcal{X}_g = \mathbb{B}^N \times \mathcal{P}(\{1,\dots,N\})^{\mathbb{N}}$

3.3.2/ UNE MÉTRIQUE POUR $\mathcal{X}_g$

Cette nouvelle distance va comparer des ensembles. On rappelle quelques notions ensemblistes. Pour A et B deux ensembles de l'univers Ω, on rappelle la définition de l'opérateur de *différence ensembliste* symétrique :

$$A \Delta B = (A \cap \overline{B}) \cup (\overline{A} \cap B)$$

où $\overline{B}$ désigne le complémentaire de B dans Ω.

On considère l'espace $\mathcal{X}_g = \mathcal{P}(\{1,\dots,N\})^{\mathbb{N}} \times \mathbb{B}^N$ et on définit la distance d entre les points $X = (S,x)$ et $X' = (S',x')$ de $\mathcal{X}_g$ par

$$d(X,X') = d_H(x,x') + d_S(S,S'), \text{ où } \begin{cases} d_H(x,x') = \sum_{i=1}^{N} |x_i - x'_i| \\ d_S(S,S') = \frac{9}{N} \sum_{t\in\mathbb{N}} \frac{|S_t \Delta S'_t|}{10^{t+1}}. \end{cases} . \tag{3.6}$$

La fonction d est une somme de deux fonctions. La fonction d_H est la distance de Hamming ; il est aussi établi que la somme de deux distances est une distance. Ainsi, pour montrer que d est aussi une distance, il suffit de montrer que d_S en est une aussi, ce qui est fait en annexe B.1.

La section suivante caractérise les fonctions f qui sont chaotiques pour le schéma généralisé.

3.3.3/ CARACTÉRISATION DES FONCTIONS RENDANT CHAOTIQUES G_{f_g} SUR $\mathcal{X}_g$

On reprend les définitions des ensembles $\mathcal{T}$, $\mathcal{R}$ et $\mathcal{C}$ en les adaptant à G_{f_g}. On a les théorèmes suivants dont les preuves sont données en annexe B.2.

Théorème 12. *G_{f_g} est transitive si et seulement si* GIG(f) *est fortement connexe.*

Théorème 13. *$\mathcal{T} \subset \mathcal{R}$.*

On peut conclure que $\mathcal{C} = \mathcal{R} \cap \mathcal{T} = \mathcal{T}$. On a alors la caractérisation suivante :

Théorème 14. *Soit $f : \mathbb{B}^N \to \mathbb{B}^N$. La fonction G_{f_g} est chaotique si et seulement si* GIG(f) *est fortement connexe.*

3.4/ GÉNÉRER DES FONCTIONS CHAOTIQUES

Cette section présente deux approches permettant de générer des fonctions f dont le graphe des itérations GIU(f) est fortement connexe. La première est algorithmique mais grossière (section 3.4.1) tandis que la seconde s'appuie sur des conditions suffisantes sur le graphe d'interactions de la fonction booléenne f (Section 3.4.2).

3.4.1/ GÉNÉRATION ALGORITHMIQUE GROSSIÈRE

Cette section présente une première approche permettant de générer une fonction booléenne f dont le graphe des itérations GIU(f) est fortement connexe. La méthode est itérative et basée sur une démarche générer-tester.

On considère en premier lieu la fonction négation $\neg$ dont le graphe des itérations GIU($\neg$) est fortement connexe. Soit un graphe GIU, initialisé avec GIU($\neg$). L'algorithme effectue itérativement les deux étapes suivantes :

1. sélection aléatoire d'un arc du graphe d'itérations en cours, GIU, puis
2. analyse de la forte connexité du graphe d'itérations en cours auquel on enlèverait cet arc. Dans le cas positif, cet arc est enlevé de GIU,

et ce jusqu'à ce qu'un taux r d'arcs enlevés soit supérieur à un seuil donné par l'utilisateur. Si r est proche de 0% (*i.e.* peu d'arcs ont été supprimés), il reste peu ou prou $n \times 2^n$ arcs. Dans le cas contraire, si r est proche de 100%, il ne reste plus qu'environ 2^n arcs. Dans tous les cas, cette étape retourne un graphe GIU qui est fortement connexe. A partir du graphe GIU, il est alors facile de construire la fonction f.

Même si cet algorithme retourne toujours des fonctions dont le graphe des itérations est fortement connexe, il n'en est pas pour le moins efficace. Un défaut de l'algorithme est de nécessiter une vérification systématique de forte connexité sur le graphe entier composé de 2^N sommets et ce, à chaque itération ! La section suivante propose une solution à ce problème. Elle présente des conditions suffisantes sur un graphe à N sommets qui permettent d'obtenir des graphes d'itérations fortement connexes.

3.4.2/ CONDITIONS SUFFISANTES SUR LE GRAPHE D'INTERACTIONS

Cette partie énonce un théorème dont les hypothèses portent sur les interactions entre x_i et f_j et qui permet de n'engendrer que des fonctions f dont le graphe d'itérations GIU(f) est fortement connexe.

Théorème [15]. *Soit f une fonction de $\mathbb{B}^N$ vers lui-même telle que :*

1. *$\Gamma(f)$ n'a pas de cycle de longueur supérieure ou égale à deux ;*
2. *chaque sommet de $\Gamma(f)$ qui possède une boucle positive a aussi une boucle négative ;*
3. *chaque sommet de $\Gamma(f)$ est accessible depuis un sommet qui possède une boucle négative.*

Alors, GIU(f) *est fortement connexe.*

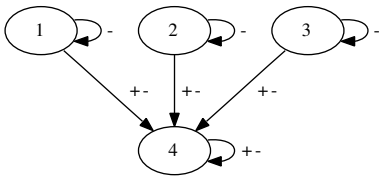


FIGURE 3.1 – Exemple de graphe d'interactions vérifiant le théorème 15

La preuve de ce théorème est donnée en annexe B.3.

Illustrons ce théorème par un exemple. On considère le graphe d'interactions $\Gamma(f)$ donné en figure 3.1. Il vérifie le théorème 15 : toutes les fonctions f possédant un tel graphe d'interactions ont un graphe d'itérations GIU(f) fortement connexe. Pratiquement, il existe 34226 fonctions de $\mathbb{B}^4$ dans lui même qui vérifient ce graphe d'intéraction. Cependant, nombreuses sont celles qui possèdent un comportement équivalent. Deux fonctions sont équivalentes si leurs GIU sont isomorphes (au sens de l'isomorphisme de graphes). Il ne reste alors plus que 520 fonctions f non équivalentes de graphe d'interactions $\Gamma(f)$.

3.5/ CONCLUSION

Ce chapitre a montré que les itérations unaires sont chaotiques si et seulement si le graphe GIU(f) est fortement connexe et que les itérations généralisées sont chaotiques si et seulement si le graphe GIG(f) est aussi fortement connexe. On dispose ainsi a priori d'une collection infinie de fonctions chaotiques. Le chapitre suivant s'intéresse à essayer de prédire le comportement de telles fonctions.

4

PRÉDICTION DES SYSTÈMES CHAOTIQUES

Les réseaux de neurones chaotiques ont été étudiés à de maintes reprises en raison de leurs applications potentielles : les composants utiles à la sécurité comme les fonctions de hachage [LDX10a], le tatouage numérique [SJT+04, ZLW05] ou les schémas de chiffrement [Lia09]. Dans tous ces cas, l'emploi de fonctions chaotiques est motivé par leur comportement imprévisible et proche de l'aléa.

Les réseaux de neurones chaotiques peuvent être conçus selon plusieurs principes. Des neurones modifiant leur état en suivant une fonction non linéaire sont par exemple appelés neurones chaotiques [CGH07]. L'architecture de réseaux de neurones de type Perceptron multi-couches (MLP) n'itèrent quant à eux classiquement pas de fonction chaotique : leurs fonctions d'activation sont usuellement choisies parmi les sigmoïdes (la fonction tangeante hyperbolique par exemple). Il a cependant été démontré que ce sont des approximateurs universels [Cyb89, HSW89]. Ils permettent, dans certains cas, de simuler des comportements physiques chaotiques comme le circuit de Chua [DD10]. Parfois [LDX10b], la fonction de transfert de cette famille de réseau et celle d'initialisation sont toutes les deux définies à l'aide de fonctions chaotiques.

Ces réseaux de neurones partagent le fait qu'ils sont qualifiés de "chaotiques" sous prétexte qu'ils embarquent une fonction de ce type et ce sans qu'aucune preuve rigoureuse ne soit fournie. Ce chapitre caractérise la classe des réseaux de neurones MLP chaotiques. Il s'intéresse ensuite à l'étude de prévisibilité de systèmes dynamiques discrets chaotiques par cette famille de MLP.

La section 4.1 définit la construction d'un réseau de neurones chaotique selon Devanay. La section 4.2 présente l'approche duale de vérification si un réseau de neurones est chaotique ou non. La section 4.3 s'intéresse à étudier pratiquement si un réseau de neurones peut approximer des itérations unaires chaotiques, ces itérations étant obtenues à partir de fonctions issues de la démarche détaillée dans le chapitre précédent. Ce travail a été publié dans [BCGS12].

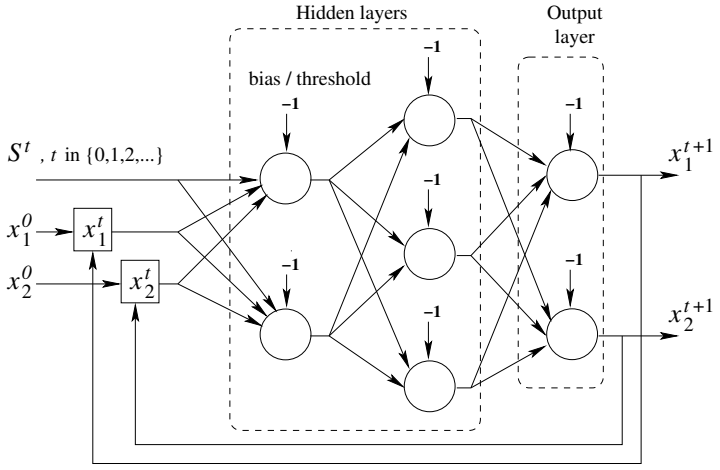


FIGURE 4.1 – Un Perceptron équivalent aux itérations unaires

4.1/ UN RÉSEAU DE NEURONES CHAOTIQUE AU SENS DE DEVANEY

On considère une fonction $f : \mathbb{B}^N \to \mathbb{B}^N$ telle que GIU(f) est fortement connexe. Ainsi G_{f_u} est chaotique d'après le théorème 11.

On considère ici le schéma unaire défini par l'équation (3.2). On construit un Perceptron multi-couches associé à la fonction F_{f_u}. Plus précisément, pour chaque entrée $(x, s) \in \mathbb{B}^N \times [N]$, la couche de sortie doit générer $F_{f_u}(x, s)$. On peut ainsi lier la couche de sortie avec celle d'entrée pour représenter les dépendances entre deux itérations successives. On obtient une réseau de neurones dont le comportement est le suivant (voir Figure. 4.1) :

— Le réseau est initialisé avec le vecteur d'entrée $\left(x^0, S^0\right) \mathbb{B}^N \times [N]$ et calcule le vecteur de sortie $x^1 = F_{f_u}\left(x^0, S^0\right)$. Ce vecteur est publié comme une sortie et est aussi retourné sur la couche d'entrée à travers les liens de retour.
— Lorsque le réseau est activé à la $t^{\text{ème}}$ itération, l'état du système $x^t \in \mathbb{B}^N$ reçu depuis la couche de sortie ainsi que le premier terme de la séquence $(S^t)^{t\in\mathbb{N}}$ (*i.e.*, $S^0 \in [N]$) servent à construire le nouveau vecteur de sortie. Ce nouveau vecteur, qui représente le nouvel état du système dynamique, satisfait :

$$x^{t+1} = F_{f_u}(x^t, S^0) \in \mathbb{B}^N \quad . \tag{4.1}$$

Le comportement de ce réseau de neurones est tel que lorsque l'état initial est composé de $x^0 \in \mathbb{B}^N$ et d'une séquence $(S^t)^{t\in\mathbb{N}}$, alors la séquence contenant les vecteurs successifs publiés $(x^t)^{t\in\mathbb{N}^*}$ est exactement celle produite par les itérations unaires décrites à la section 3.2. Mathématiquement, cela signifie que si on utilise les mêmes vecteurs d'entrée les deux approches génèrent successivement les mêmes sorties. En d'autres termes ce réseau de neurones modélise le comportement de G_{f_u}, dont les itérations sont chaotiques sur $\mathcal{X}_u$. On peut donc le qualifier de chaotique au sens de Devaney.

4.2/ VÉRIFIER SI UN RÉSEAU DE NEURONES EST CHAOTIQUE

On s'intéresse maintenant au cas où l'on dispose d'un réseau de neurones de type Perceptron multi-couches dont on cherche à savoir s'il est chaotique (parce qu'il a par exemple été déclaré comme tel) au sens de Devaney. On considère de plus que sa topologie est la suivante : l'entrée est constituée de N bits et un entier, la sortie est constituée de N bits et chaque sortie est liée à une entrée par une boucle.

— Le réseau est initialisé avec N bits $\left(x_1^0, \ldots, x_{\mathsf{N}}^0\right)$ et une valeur entière $S^0 \in [\mathsf{N}]$.
— A l'itération t, le vecteur $\left(x_1^t, \ldots, x_{\mathsf{N}}^t\right)$ permet de construire les N bits servant de sortie $\left(x_1^{t+1}, \ldots, x_{\mathsf{N}}^{t+1}\right)$.

Le comportement de ce type de réseau de neurones peut être prouvé comme étant chaotique en suivant la démarche suivante. On nomme tout d'abord $F : \mathbb{B}^{\mathsf{N}} \times [\mathsf{N}] \to \mathbb{B}^{\mathsf{N}}$ la fonction qui associe au vecteur $((x_1, \ldots, x_{\mathsf{N}}), s) \in \mathbb{B}^{\mathsf{N}} \times [\mathsf{N}]$ le vecteur $(y_1, \ldots, y_{\mathsf{N}}) \in \mathbb{B}^{\mathsf{N}}$, où $(y_1, \ldots, y_{\mathsf{N}})$ sont les sorties du réseau neuronal après l'initialisation de la couche d'entrée avec $(s, (x_1, \ldots, x_{\mathsf{N}}))$. Ensuite, on définie $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ telle que $f(x_1, x_2, \ldots, x_{\mathsf{N}})$ est égal à

$$\left(F\left((x_1, x_2, \ldots, x_{\mathsf{N}}), 1\right), \ldots, F\left((x_1, x_2, \ldots, x_{\mathsf{N}}), \mathsf{N}\right)\right). \tag{4.2}$$

Ainsi pour chaque j, $j \in [\mathsf{N}]$, on a $f_j(x_1, x_2, \ldots, x_{\mathsf{N}}) = F\left((x_1, x_2, \ldots, x_{\mathsf{N}}), j\right)$. Si ce réseau de neurones est initialisé avec $\left(x_1^0, \ldots, x_{\mathsf{N}}^0\right)$ et $S \in [\mathsf{N}]^{\mathbb{N}}$, il produit exactement les mêmes sorties que les itérations de F_{f_u} avec une condition initiale $\left((x_1^0, \ldots, x_{\mathsf{N}}^0), S\right) \in \mathbb{B}^{\mathsf{N}} \times [\mathsf{N}]^{\mathbb{N}}$. Les itérations de F_{f_u} sont donc un modèle formel de cette classe de réseaux de neurones. Pour vérifier si un de ces représentants est chaotique, il suffit ainsi de vérifier si le graphe d'itérations $\mathrm{GIU}(f)$ est fortement connexe.

4.3/ UN RÉSEAU DE NEURONES PEUT-IL APPROXIMER DES ITÉRATION UNAIRES CHAOTIQUES ?

Cette section s'intéresse à étudier le comportement d'un réseau de neurones face à des itérations unaires chaotiques, comme définies à la section 3.2. Plus précisément, on considère dans cette partie une fonction dont le graphe des itérations unaires est fortement connexe et une séquence dans $[\mathsf{N}]^{\mathbb{N}}$. On cherche à construire un réseau de neurones qui approximerait les itérations de la fonction G_{f_u} comme définie à l'équation (3.3).

Sans perte de généralité, on considère dans ce qui suit une instance de fonction à quatre éléments.

4.3.1/ CONSTRUCTION DU RÉSEAU

On considère par exemple les deux fonctions f et g de $\mathbb{B}^4$ dans $\mathbb{B}^4$ définies par :

$$\begin{aligned} f(x_1, x_2, x_3, x_4) &= (x_1(x_2 + x_4) + \overline{x_2}x_3\overline{x_4}, x_2, x_3(\overline{x_1}.\overline{x_4} + x_2x_4 + x_1\overline{x_2}), x_4 + \overline{x_2}x_3) \\ g(x_1, x_2, x_3, x_4) &= (\overline{x_1}, \overline{x_2} + x_1.\overline{x_3}.\overline{x_4}, \overline{x_3}(x_1 + x_2 + x_4), \overline{x_4}(x_1 + \overline{x_2} + \overline{x_3})) \end{aligned}$$

On peut vérifier facilement que le graphe GIU(f) n'est pas fortement connexe car $(1,1,1,1)$ est un point fixe de f tandis que le graphe GIU(g) l'est.

L'entrée du réseau est une paire de la forme $(x,(S^t)^{t\in\mathbb{N}})$ et sa sortie correspondante est de la forme $\left(F_{h_u}(S^0,x),\sigma((S^t)^{t\in\mathbb{N}})\right)$ comme définie à l'équation (3.3).

On s'intéresse d'abord aux différentes manières de mémoriser des configurations. On en considère deux principalement. Dans le premier cas, on considère une entrée booléenne par élément tandis que dans le second cas, les configurations sont mémorisées comme des entiers naturels. Dans ce dernier cas, une approche naïve pourrait consister à attribuer à chaque configuration de $\mathbb{B}^N$ l'entier naturel correspondant. Cependant, une telle représentation rapproche arbitrairement des configurations diamétralement opposées dans le N-cube comme une puissance de deux et la configuration immédiatement précédente : 10000 serait modélisée par 16 et et 01111 par 15 alors que leur distance de Hamming est 15. De manière similaire, ce codage éloigne des configurations qui sont très proches : par exemple 10000 et 00000 ont une distance de Hamming de 1 et sont respectivement représentées par 16 et 0. Pour ces raisons, le codage retenu est celui des codes de Gray [Gra53].

Concentrons nous sur la traduction de la stratégie. Il n'est naturellement pas possible de traduire une stratégie infinie quelconque à l'aide d'un nombre fini d'éléments. On se restreint donc à des stratégies de taille l, $2 \leq l \leq k$, où k est un paramètre défini initialement. Chaque stratégie est mémorisée comme un entier naturel exprimé en base $N+1$: à chaque itération, soit aucun élément n'est modifié, soit un élément l'est. Enfin, on donne une dernière entrée : $m \in [l-1]$, qui est le nombre d'itérations successives que l'on applique en commençant à x. Les sorties (stratégies et configurations) sont mémorisées selon les mêmes règles.

Concentrons nous sur la complexité du problème. Chaque entrée de l'entrée-sortie de l'outil est un triplet composé d'une configuration x, d'un extrait S de la stratégie à itérer de taille l, $2 \leq l \leq k$ et d'un nombre $m \in [l-1]$ d'itérations à exécuter. Il y a 2^N configurations x et N^l stratégies de taille l. De plus, pour une configuration donnée, il y a $\omega = 1\times N^2 + 2\times N^3 + \ldots + (k-1)\times N^k$ manières d'écrire le couple (m,S). Il n'est pas difficile d'établir que

$$(N-1)\times\omega = (k-1)\times N^{k+1} - \sum_{i=2}^{k} N^i$$

donc

$$\omega = \frac{(k-1)\times N^{k+1}}{N-1} - \frac{N^{k+1}-N^2}{(N-1)^2} \quad .$$

Ainsi le nombre de paires d'entrée-sortie pour les réseaux de neurones considérés est

$$2^N \times \left(\frac{(k-1)\times N^{k+1}}{N-1} - \frac{N^{k+1}-N^2}{(N-1)^2}\right) \quad .$$

Par exemple, pour 4 éléments binaires et une stratégie d'au plus 3 termes on obtient 2304 couples d'entrée-sortie.

4.3.2/ EXPÉRIMENTATIONS

On se focalise dans cette section sur l'entraînement d'un MLP pour apprendre des itérations chaotiques. Ce type de réseau ayant déjà été évalué avec succès dans

la prédiction de séries chaotiques temporelles. En effet, les auteurs de [DD10] ont montré qu'un MLP pouvait apprendre la dynamique du circuit de Chua. Ce réseau avec rétropropagation est composé de deux couches et entraîné à l'aide d'une propagation arrière Bayesienne.

Les choix de l'architecture de réseau ainsi que ceux concernant les méthodes d'apprentissage ont été détaillés dans [BCGS12]. En pratique, nous avons considéré des configurations de quatre éléments booléens et une stratégie fixe de longueur 3. Pour le premier codage, nous avons ainsi 6 entrées et 5 sorties tandis que pour le second, uniquement 3 entrées et 2 sorties. Cela engendre ainsi 2304 combinaisons possibles comme détaillé à la section précédente.

Topologie du réseau : 6 entrées, 5 sorties, 1 couche cachée				
Neurones cachés		10 neurones		
Epochs		125	250	500
Chaotique g	Sortie (1)	90.92%	91.75%	91.82%
	Sortie (2)	69.32%	78.46%	82.15%
	Sortie (3)	68.47%	78.49%	82.22%
	Sortie (4)	91.53%	92.37%	93.4%
	Config.	36.10%	51.35%	56.85%
	Stratégie (5)	1.91%	3.38%	2.43%
Non-chaotic f	Sortie (1)	97.64%	98.10%	98.20%
	Sortie (2)	95.15%	95.39%	95.46%
	Sortie (3)	100%	100%	100%
	Sortie (4)	97.47%	97.90%	97.99%
	Config.	90.52%	91.59%	91.73%
	Stratégie (5)	3.41%	3.40%	3.47%
Neurones cachés		25 neurones		
Epochs		125	250	500
Chaotique g	Sortie (1)	91.65%	92.69%	93.93%
	Sortie (2)	72.06%	88.46%	90.5%
	Sortie (3)	79.19%	89.83%	91.59%
	Sortie (4)	91.61%	92.34%	93.47%
	Config.	48.82%	67.80%	70.97%
	Stratégie (5)	2.62%	3.43%	3.78%
Non-chaotique f	Sortie (1)	97.87%	97.99%	98.03%
	Sortie (2)	95.46%	95.84%	96.75%
	Sortie (3)	100%	100%	100%
	Sortie (4)	97.77%	97.82%	98.06%
	Config.	91.36%	91.99%	93.03%
	Stratégie (5)	3.37%	3.44%	3.29%

TABLE 4.1 – Taux de prédiction lorsque les configurations sont exprimées comme un vecteur booléen.

Le tableau 4.1 synthétise les résultats obtenus avec le premier codage. Sans surprise, la précision de la prédiction croit avec l'Epoch et le nombre de neurones sur la couche cachée. Dans tous les cas, les résultats sont plus précis dans le cas non chaotique que dans l'autre. Enfin, le réseau ne parvient jamais à apprendre le comportement de la stratégie.

Les résultats concernant le second codage (*i.e.*, avec les codes de Gray) sont synthétisés dans le tableau 4.2. On constate que le réseau apprend cinq fois mieux les comportements non chaotiques que ceux qui le sont. Ceci est illustré au travers des figures 4.2(a)

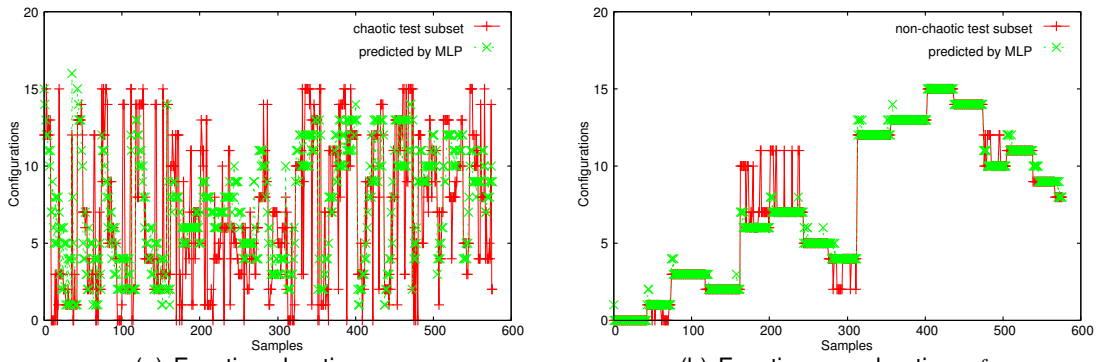
et 4.2(b). De plus, comme dans le codage précédent, les stratégies ne peuvent pas être prédites. On constate que ce second codage réduit certes le nombre de sorties, mais est largement moins performant que le premier. On peut expliquer ceci par le fait que ce second codage garantit que deux entiers successifs correspondent à deux configurations voisines, *i.e.*, qui ne diffèrent que d'un élément. La réciproque n'est cependant pas établie et deux configurations voisines peuvent être traduites par des entiers très éloignés et ainsi difficiles à apprendre.

4.4/ CONCLUSION

Dans ce chapitre, nous avons établi une similitude entre les itérations chaotiques et une famille de Perceptrons multi-couches. Nous avons d'abord montré comment construire un réseau de neurones ayant un comportement chaotique. Nous avons présenté ensuite comment vérifier si un réseau de neurones établi était chaotique. Nous avons enfin montré en pratique qu'il est difficile pour un réseau de neurones d'apprendre le comportement global d'itérations chaotiques. Comme il est difficile (voir impossible) d'apprendre le comportement de telles fonctions, il paraît naturel de savoir si celles ci peuvent être utilisées pour générer des nombres pseudo-aléatoires, ce que propose la partie suivante.

Topologie du réseau : 3 entrées, 2 sorties, 1 couche cachée				
	Neurones cachés	10 neurones		
	Epochs	125	250	500
Chaotique g	Config. (1)	13.29%	13.55%	13.08%
	Stratégie (2)	0.50%	0.52%	1.32%
Non-Chaotique f	Config. (1)	77.12%	74.00%	72.60%
	Stratégie (2)	0.42%	0.80%	1.16%
	Neurones cachés	25 neurones		
	Epochs	125	250	500
Chaotique g	Config. (1)	12.27%	13.15%	13.05%
	Stratégie (2)	0.71%	0.66%	0.88%
Non-Chaotique f	Config. (1)	73.60%	74.70%	75.89%
	Stratégie (2)	0.64%	0.97%	1.23%

TABLE 4.2 – Taux de prédiction lorsque les configurations sont exprimées à l'aide de codes de Gray.



(a) Fonction chaotique g (b) Fonction non-chaotique f

FIGURE 4.2 – Prédiction lorsque les configurations sont exprimées à l'aide de codes de Gray.

III

APPLICATIONS À LA GÉNÉRATION DE NOMBRES PSEUDO-ALÉATOIRES

5

CARACTÉRISATION DES GÉNÉRATEURS CHAOTIQUES

Au bout d'un nombre b d'itérations, si la fonction, notée G_{f_u} (ou bien G_{f_g}) présentée au chapitre 3, a de « bonnes » propriétés chaotiques, le mot x^b devrait « sembler ne plus dépendre » de x^0. On peut penser à exploiter une de ces fonctions G_f comme un générateur aléatoire.

Ce chapitre présente donc une application directe de la théorie développée ci-avant à la génération de nombres pseudo-aléatoires. La section 5.1 présente un état de l'art (incomplet) de l'exploitation de fonctions au comportement chaotique pour obtenir des PRNGs. La section 5.2 présente ensuite l'algorithme de PRNG. La contrainte de distribution uniforme de la sortie est discutée dans cette section. La chaoticité du générateur est étudiée en section 5.3. La section 5.2 a été publiée à [BCGW11, BCGR11].

5.1/ QUELQUES PRNGS BASÉS SUR DES FONCTIONS AUX ITÉRATIONS CHAOTIQUES

Les PRNGs chaotiques (CPRNGs) sont des générateurs non linéaires définis par $x_0 \in \mathbb{R}$ et $x_{t+1} = f(x_t)$, où f est une fonction au comportement chaotique. Les raisons qui expliquent l'intérêt de telles fonctions sont naturellement la sensibilité aux conditions initiales, leur imprévisibilité... Cependant, comme l'ordinateur sur lequel elles s'exécutent a une précision finie, les générateurs qui les embarquent peuvent avoir des périodes arbitrairement courtes, ne pas fournir de sortie selon une distribution uniforme... D'un point de vue cryptographique, ces CPRNGs sont critiquables [Wig03]. Réduire ces critiques est l'objectif de nombreux travaux de recherche reportés ci dessous.

Parmi les suites simples classiquement embarquées dans les CPRNGs, on trouve principalement la suite logistique, la suite de Hénon. La suite logistique [M+76] est définie de $[0;1]$ dans lui même par $x_{t+1} = r \times x_t(1 - x_t)$ avec $x_0 \in [0;1]$ et $3,57 < r < 4,0$. La suite de Hénon [Hén76] de $A \times B$ dans lui même, avec A et B deux sous-ensembles de $\mathbb{R}$, est définie par $x_{t+1} = (1 - ax_t^2) + y_t$ et $y_{t+1} = bx_{t+1}$, où a et b sont les paramètres canoniques. Pour $a = 1,4$, $b = 0,3$, $A = [-1,5;1,5]$ et $B = -[0,4;0,4]$ le comportement de cette suite est chaotique.

La suite logistique est utilisée dans l'article [DP11] dans lequel les auteurs utilisent une représentation des réels à virgule fixe. Les auteurs de [DP12] comparent leur implantation

de la suite logistique avec celle de la suite de Hénon. Les auteurs de [DP14] ont exploité la réécriture de la suite logistique sous la forme $x_{t+1} = (r \times x_t) - (r \times x_t^2)$ et la possibilité de paralléliser ceci pour accroître la fréquence du PRNG. Les auteurs de [LSXC08] proposent de coupler deux suites logistiques, chacune étant paramétrée différemment (x_0, r_1 et y_0, r_2 respectivement). L'idée principale est de modifier iterrativement le paramètre r_2 à l'aide des itérés de x_t. Quant aux auteurs de [MAPS13], ils couplent la suite logistique et celle de Hénon. La première suite sert à sélectionner les éléments parmi ceux générés par la seconde. Les auteurs de [MCL06], combinent spatialement L suites logistiques et construisent ainsi $x_t(0), \ldots, x_t(L-1)$ selon l'équation suivante :

$$x_{t+1}(i) = (1-\varepsilon) f(x_t(i)) + \frac{\varepsilon}{2}(f(x_t(i-1)) + f(x_t(i+1))), \tag{5.1}$$

où $i \in \frac{\mathbb{Z}}{L\mathbb{Z}}$, f est une adaptation de la suite logistique au cas discret, la graine $(X_0(0), \ldots, X_0(L-1))$ et la pondération ε sont fournies par l'utilisateur.

René Lozi a aussi étudié la construction de PRNGs en couplant des suites de Lozi [ERTL11] (qui sont une variation des suites de Hénon : x_t^2 est remplacé par $|x_t|$), la suite tente [Loz12] et en extrayant des sous-suites pour construire la sortie du PRNG [LT14].

Certaines équations différentielles ont été à la base de PRNGs chaotiques. On pense aux équations de Lorenz [Lor63], à celles de Rössler [Rö76]... Celles-ci ont par exemple embarquées dans les PRNG dans les articles [SCN09] et [MBZ+13] respectivement.

5.2/ NOMBRES PSEUDO-ALÉATOIRES CONSTRUITS PAR ITÉRATIONS UNAIRES

Input : une fonction f, un nombre d'itérations b, une configuration initiale x^0 (N bits)
Output : une configuration x (N bits)
$x \leftarrow x^0$;
for $i = 1, \ldots, b$ **do**
  $s \leftarrow$ *Random*(N);
  $x \leftarrow F_{f_u}(x, s)$;
end
return x;

Algorithme 1 : PRNG basé sur les itérations unaires.

5.2.1/ ALGORITHME D'UN GÉNÉRATEUR

On peut penser à construire un générateur de nombres pseudo-aléatoires comme dans l'algorithme 1 donné ci-dessous.

Celui-ci prend en entrée : une fonction f ; un entier b, qui est le nombre d'itérations à effectuer entre deux sorties et une configuration initiale x^0. Il retourne une nouvelle configuration x en appliquant la fonction F_{f_u} (équation 3.1 vue au chapitre 3) et correspondant à des itérations unaires. En interne, il exploite un algorithme de génération de nombres

pseudo-aléatoires donné en paramètre. Cela peut être n'importe quel PRNG (XORshift, Mersenne-Twister) dont la sortie est uniformément distribuée. Notre approche vise a donner des propriétés de chaos à ce générateur embarqué.

Nous avons vu au chapitre 3 que G_{f_u} est chaotique dans l'espace $\mathcal{X}_u$ si et seulement si le graphe d'itérations GIU(f) est fortement connexe. Pour $b = 1$, l'algorithme itère la fonction F_{f_u}.

Pour simuler au mieux l'aléa, un bon générateur de nombres pseudo-aléatoires se doit de fournir des nombres selon une distribution uniforme. Regardons comment l'uniformité de la distribution contraint la fonction f à itérer.

5.2.2/ UN GÉNÉRATEUR À SORTIE UNIFORMÉMENT DISTRIBUÉE

Une matrice stochastique est une matrice carrée dont tous les éléments sont positifs ou nuls et dont la somme de chaque ligne vaut 1. Une matrice stochastique l'est doublement si la somme de chaque colonne est 1. Enfin, une matrice stochastique de taille $n \times n$ est régulière si la propriété suivante est établie :

$$\exists k \in \mathbb{N}^*, \forall i, j \in [\![1; n]\!], M_{ij}^k > 0.$$

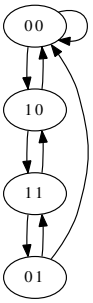
On énonce le théorème classique suivant liant les vecteurs de probabilités et les chaînes de Markov.

Théorème [16]. *Si M est une matrice stochastique régulière, alors M possède un unique vecteur stationnaire de probabilités π ($\pi.M = \pi$). De plus, si π^0 est un vecteur de probabilités et si on définit la suite $(\pi^k)^{k\in\mathbb{N}}$ par $\pi^{k+1} = \pi^k.M$ pour $k = 0, 1, \ldots$ alors la chaîne de Markov π^k converge vers π lorsque k tend vers l'infini.*

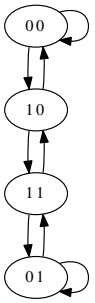
Montrons sur un exemple jouet à deux éléments que ce théorème permet de vérifier si la sortie d'un générateur de nombres pseudo-aléatoires est uniformément distribuée ou non. Soient alors g et h deux fonctions de $\mathbb{B}^2$ définies par $g(x_1, x_2) = (\overline{x_1}, x_1.\overline{x_2})$ et $h(x_1, x_2) = (\overline{x_1}, x_1\overline{x_2} + \overline{x_1}x_2)$. Leurs graphes d'interactions donnés en figure 5.2(a) et 5.2(b) vérifient les hypothèses du théorème 15. Leurs graphes d'itérations sont donc fortement connexes, ce que l'on peut vérifier aux figures 5.1(a) et 5.1(b). *A priori*, ces deux fonctions pourraient être intégrées dans un générateur de nombres pseudo-aléatoires. Montrons que ce n'est pas le cas pour g et que cela l'est pour h.

Comme le générateur *Random* possède une sortie uniformément distribuée, la stratégie est uniforme sur $[\![1, 2]\!]$, et donc, pour tout sommet de GIU(g) et de GIU(h), chaque arc sortant de ce sommet a, parmi l'ensemble des arcs sortant de ce sommet, une probabilité $1/2$ d'être celui qui sera traversé. En d'autres mots, GIU(g) est le graphe orienté d'une chaîne de Markov. Il est facile de vérifier que la matrice de transitions d'un tel processus est $M_g = \frac{1}{2}\check{M}_g$, où $\check{M}_g$ est la matrice d' adjacence donnée en figure 5.3(a) (voir ci-après), et de manière similaire pour M_h.

Les deux matrices M_g et M_h sont stochastiques. Pour montrer qu'elles sont régulières il suffit de constater qu'aucun élément de M_g^5 ni de M_h^3 n'est nul. De plus, les vecteurs de probabilités $\pi_g = (\frac{4}{10}, \frac{1}{10}, \frac{3}{10}, \frac{2}{10})$ et $\pi_h = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ vérifient $\pi_g M_g = \pi_g$ et $\pi_h M_h = \pi_h$. Alors d'après le théorème 16, pour n'importe quel vecteur initial de probabilités π^0, on a $\lim_{k\to\infty} \pi^0 M_g^k = \pi_g$ et $\lim_{k\to\infty} \pi^0 M_h^k = \pi_h$. Ainsi la chaîne de Markov associée à h tend



(a) $g(x_1, x_2) = (\overline{x_1}, x_1\overline{x_2})$



(b) $h(x_1, x_2) = (\overline{x_1}, x_1\overline{x_2} + \overline{x_1}x_2)$

FIGURE 5.1 – Graphes des itérations unaires de fonctions booléennes dans $\mathbb{B}^2$

vers une distribution uniforme, contrairement à celle associée à g. On en déduit que g ne devrait pas être itérée dans un générateur de nombres pseudo-aléatoires. Au contraire, h devrait pouvoir être embarquée dans l'algorithme 1, pour peu que le nombre b d'itérations entre deux mesures successives de valeurs soit suffisamment grand de sorte que le vecteur d'état de la chaîne de Markov ait une distribution suffisamment proche de la distribution uniforme.

On énonce directement le théorème suivant dont la preuve est donnée en annexe C.

Théorème 17 (Uniformité de la sortie de l'algorithme 1)**.** *Soit* $f : \mathbb{B}^N \to \mathbb{B}^N$, GIU($f$) *son graphe d'itérations ,* $\check{M}$ *sa matrice d'adjacence et* M *une matrice* $2^n \times 2^n$ *définie par* $M = \frac{1}{N}\check{M}$*. Si* GIU(f) *est fortement connexe, alors la sortie du générateur de nombres pseudo-aléatoires détaillé par l'algorithme 1 suit une loi qui tend vers la distribution uniforme si et seulement si* M *est une matrice doublement stochastique.*

5.2.3/ QUELQUES EXEMPLES

On considère le graphe d'interactions $\Gamma(f)$ donné en figure 5.4(a). C'est le même qui a été présenté à la section 3.4. On a vu qu'il y avait 520 fonctions f non isomorphes de graphe d'interactions $\Gamma(f)$.

Seulement 16 d'entre elles possèdent une matrice doublement stochastique. La figure 5.4(b) explicite ces 16 fonctions en définissant les images des éléments de la liste 0, 1, 2,. . . , 14, 15 en respectant l'ordre. Expliquons enfin comment a été calculé le nombre de la troisième colonne utilisé comme le paramètre b dans l'algorithme 1.

Soit e_i le $i^{\text{ème}}$ vecteur de la base canonique de $\mathbb{R}^{2^N}$. Chacun des éléments v_j, $1 \leq j \leq 2^N$, du vecteur $e_i M_f^t$ représente la probabilité d'être dans la configuration j après t étapes du processus de Markov associé à GIU(f) en partant de la configuration i. Le nombre $\min\{t \mid t \in \mathbb{N}, \left\|e_i M_f^t - \pi\right\|_2 < 10^{-4}\}$ représente le plus petit nombre d'itérations où la distance de ce vecteur au vecteur $\pi = (\frac{1}{2^N}, \ldots, \frac{1}{2^N})$ – autrement dit, où la déviation par rapport à la distribution uniforme – est inférieure à 10^{-4}. En prenant le max pour tous les e_i, on obtient une valeur pour b. Ainsi, on a

$$b = \max_{i \in [\![1, 2^N]\!]} \left\{ \min \left\{ t \mid t \in \mathbb{N}, \left\|e_i M_f^t - \pi\right\|_2 < 10^{-4} \right\} \right\}. \tag{5.2}$$

(a) $g(x_1, x_2) = (\overline{x_1}, x_1\overline{x_2})$ (b) $h(x_1, x_2) = (\overline{x_1}, x_1\overline{x_2} + \overline{x_1}x_2)$

FIGURE 5.2 – Graphes d'interactions de fonctions booléennes dans $\mathbb{B}^2$

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

(a) $\check{M}_g$. (b) $\check{M}_h$.

FIGURE 5.3 – Graphe des fonctions candidates avec $n = 2$

Par la suite, ce nombre sera appelé *temps de mélange*.

La qualité des séquences aléatoires a été évaluée à travers la suite de tests statistiques développée pour les générateurs de nombres pseudo-aléatoires par le *National Institute of Standards and Technology* (NIST). L'expérience a montré notamment que toutes ces fonctions passent avec succès cette batterie de tests.

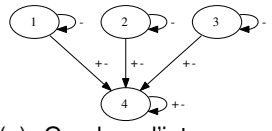
Pour conclure cette section, on remarque que le générateur de nombres pseudo-aléatoires a été prouvé chaotique pour $b = 1$, *i.e.*, lorsqu'il y a une sortie pour chaque itération. Ceci est difficilement compatible avec la volonté d'avoir une sortie uniformément distribuée : se rapprocher de cette distribution nécessite en effet un nombre plus élevé d'itérations b entre chaque sortie. Par exemple, dans l'exemple précédent, il est nécessaire d'itérer au moins 42 fois entre chaque sortie pour suivre une loi uniforme à 10^{-4} près. Montrer que les sous-séquences de suites chaotiques ainsi générées demeurent chaotiques est l'objectif de la section suivante.

5.3/ UN PRNG BASÉ SUR DES ITÉRATIONS UNAIRES QUI EST CHAOTIQUE

Cette section présente un espace métrique adapté au générateur de nombres pseudo-aléatoires présenté à l'algorithme 1 et prouve ensuite que la fonction qu'il représente est chaotique sur cet espace.

5.3.1/ UN ESPACE $\mathcal{X}_{N,\mathcal{P}}$ POUR LE PRNG DE L'ALGORITHME 1

Introduisons tout d'abord $\mathcal{P} \subset \mathbb{N}$ un ensemble fini non vide de cardinalité $p \in \mathbb{N}^*$. Intuitivement, c'est le nombre d'itérations qu'il est autorisé de faire. On ordonne les p éléments



(a) Graphe d'interactions

Nom	Définition	b
$\mathcal{F}_1$	14, 15, 12, 13, 10, 11, 8, 9, 6, 7, 4, 5, 2, 3, 1, 0	206
$\mathcal{F}_2$	14, 15, 12, 13, 10, 11, 8, 9, 6, 7, 5, 4, 3, 2, 0, 1	94
$\mathcal{F}_3$	14, 15, 12, 13, 10, 11, 8, 9, 6, 7, 5, 4, 3, 2, 1, 0	69
$\mathcal{F}_4$	14, 15, 12, 13, 10, 11, 9, 8, 6, 7, 5, 4, 3, 2, 0, 1	56
$\mathcal{F}_5$	14, 15, 12, 13, 10, 11, 9, 8, 6, 7, 5, 4, 3, 2, 1, 0	48
$\mathcal{F}_6$	14, 15, 12, 13, 10, 11, 9, 8, 7, 6, 4, 5, 2, 3, 0, 1	86
$\mathcal{F}_7$	14, 15, 12, 13, 10, 11, 9, 8, 7, 6, 4, 5, 2, 3, 1, 0	58
$\mathcal{F}_8$	14, 15, 12, 13, 10, 11, 9, 8, 7, 6, 4, 5, 3, 2, 1, 0	46
$\mathcal{F}_9$	14, 15, 12, 13, 10, 11, 9, 8, 7, 6, 5, 4, 3, 2, 0, 1	42
$\mathcal{F}_{10}$	14, 15, 12, 13, 10, 11, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0	69
$\mathcal{F}_{11}$	14, 15, 12, 13, 11, 10, 9, 8, 7, 6, 5, 4, 2, 3, 1, 0	58
$\mathcal{F}_{12}$	14, 15, 13, 12, 11, 10, 8, 9, 7, 6, 4, 5, 2, 3, 1, 0	35
$\mathcal{F}_{13}$	14, 15, 13, 12, 11, 10, 8, 9, 7, 6, 4, 5, 3, 2, 1, 0	56
$\mathcal{F}_{14}$	14, 15, 13, 12, 11, 10, 8, 9, 7, 6, 5, 4, 3, 2, 1, 0	94
$\mathcal{F}_{15}$	14, 15, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 0, 1	86
$\mathcal{F}_{16}$	14, 15, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0	206

(b) Fonctions doublement stochastiques

FIGURE 5.4 – Candidates pour le générateur avec $n = 4$

de $\mathcal{P}$ comme suit : $\mathcal{P} = \{p_1, p_2, \ldots, p_\mathsf{p}\}$ et $p_1 < p_2 < \ldots < p_\mathsf{p}$.

Dans l'algorithme 1, p vaut 1 et $p_1 = b$. Cet algorithme peut être vu comme b compostions de la fonction F_{f_u}. Ceci peut cependant se généraliser à p_i, $p_i \in \mathcal{P}$, compositions fonctionnelles de F_{f_u}. Ainsi, pour chaque $p_i \in \mathcal{P}$, on construit la fonction $F_{f_u,p_i} : \mathbb{B}^\mathsf{N} \times [\mathsf{N}]^{p_i} \to \mathbb{B}^\mathsf{N}$ définie par

$$F_{f_u,p_i}(x,(u^0,u^1,\ldots,u^{p_i-1})) = F_{f_u}(\ldots(F_{f_u}(F_{f_u}(x,u^0),u^1),\ldots),u^{p_i-1}).$$

On construit l'espace $\mathcal{X}_{\mathsf{N},\mathcal{P}} = \mathbb{B}^\mathsf{N} \times \mathbb{S}_{\mathsf{N},\mathcal{P}}$, où $\mathbb{S}_{\mathsf{N},\mathcal{P}} = [\mathsf{N}]^\mathbb{N} \times \mathcal{P}^\mathbb{N}$. Chaque élément de l'espace est une paire où le premier élément est un N-uplet de $\mathbb{B}^\mathsf{N}$ (comme dans $\mathcal{X}_u$). Le second élément est une paire $((u^k)_{k\in\mathbb{N}},(v^k)_{k\in\mathbb{N}})$ de suites infinies. La suite $(v^k)_{k\in\mathbb{N}}$ définit combien d'itérations sont exécutées au temps k entre deux sorties. La séquence $(u^k)_{k\in\mathbb{N}}$ définit quel élément est modifié (toujours au temps k).

Définissons la fonction de décalage Σ pour chaque élément de $\mathbb{S}_{\mathsf{N},\mathcal{P}}$.

$$\begin{array}{rccc} \Sigma : & \mathbb{S}_{\mathsf{N},\mathcal{P}} & \longrightarrow & \mathbb{S}_{\mathsf{N},\mathcal{P}} \\ & \left((u^k)_{k\in\mathbb{N}},(v^k)_{k\in\mathbb{N}}\right) & \longmapsto & \left(\sigma^{v^0}\left((u^k)_{k\in\mathbb{N}}\right),\sigma\left((v^k)_{k\in\mathbb{N}}\right)\right). \end{array}$$

En d'autres termes, Σ reçoit deux suites u et v et effectue v^0 décalage vers la droite sur la première et un décalage vers la droite sur la seconde.

Ainsi, les sorties $(y^0,y^1,\ldots)$ produites par le générateur détaillé dans l'algorithme 1 sont les premiers composants des itérations $X^0 = (x^0,(u,v))$ et $\forall n \in \mathbb{N}, X^{n+1} = G_{f_u,\mathcal{P}}(X^\mathsf{N})$ dans $\mathcal{X}_{\mathsf{N},\mathcal{P}}$ où $G_{f_u,\mathcal{P}}$ est définie par :

$$\begin{array}{rccc} G_{f_u,\mathcal{P}} : & \mathcal{X}_{\mathsf{N},\mathcal{P}} & \longrightarrow & \mathcal{X}_{\mathsf{N},\mathcal{P}} \\ & (e,(u,v)) & \longmapsto & \left(F_{f,v^0}\left(e,(u^0,\ldots,u^{v^0-1}\right),\Sigma(u,v)\right). \end{array} \tag{5.3}$$

5.3.2/ UNE DISTANCE SUR $\mathcal{X}_{\mathsf{N},\mathcal{P}}$

On définit la fonction d sur $\mathcal{X}_{\mathsf{N},\mathcal{P}}$ comme suit : Soit $x = (e, s)$ et $\check{x} = (\check{e}, \check{s})$ dans $\mathcal{X}_{\mathsf{N},\mathcal{P}} = \mathbb{B}^{\mathsf{N}} \times \mathbb{S}_{\mathsf{N},\mathcal{P}}$, où $s = (u, v)$ et $\check{s} = (\check{u}, \check{v})$ sont dans $\mathbb{S}_{\mathsf{N},\mathcal{P}} = \mathcal{S}_{[\![1,\mathsf{N}]\!]} \times \mathcal{S}_{\mathcal{P}}$.

1. e et $\check{e}$ sont des entiers appartenant à $[\![0, 2^{\mathsf{N}-1}]\!]$. La distance de Hamming $d_{\mathbb{B}^{\mathsf{N}}}$ entre les décompositions binaires de e et de $\check{e}$ (*i.e.*, le le nombre de bits qu'elles ont de différent) constitue la partie entière de $d(X, \check{X})$.
2. la partie décimale est construite à partir des différences entre v^0 et $\check{v}^0$, suivie des différences entre les séquences finies $u^0, u^1, \ldots, u^{v^0-1}$ et $\check{u}^0, \check{u}^1, \ldots, \check{u}^{\check{v}^0-1}$, suivie par les différences entre v^1 et $\check{v}^1$, suivie par les différences entre $u^{v^0}, u^{v^0+1}, \ldots, u^{v^1-1}$ et $\check{u}^{\check{v}^0}, \check{u}^{\check{v}^0+1}, \ldots, \check{u}^{\check{v}^1-1}$, etc.

   Plus précisément, soit $p = \lfloor \log_{10}(\max \mathcal{P}) \rfloor + 1$ et $n = \lfloor \log_{10}(\mathsf{N}) \rfloor + 1$.

   1. Les p premiers éléments de $d(x, \check{x})$ sont $|v^0 - \check{v}^0|$ écrits en base 10 et sur p indices ;
   2. les $n \times \max(\mathcal{P})$ éléments suivants servent à évaluer de combien $u^0, u^1, \ldots, u^{v^0-1}$ diffère de $\check{u}^0, \check{u}^1, \ldots, \check{u}^{\check{v}^0-1}$. Les n premiers éléments sont $|u^0 - \check{u}^0|$. Il sont suivis de $|u^1 - \check{u}^1|$ écrits à l'aide de n éléments, etc.
      1. Si $v^0 = \check{v}^0$, alors le processus se continue jusqu'à $|u^{v^0-1} - \check{u}^{\check{v}^0-1}|$ et la partie décimale de $d(X, \check{X})$ est complétée par des 0 jusqu'à atteindre $p+n\times\max(\mathcal{P})$ éléments.
      2. Si $v^0 < \check{v}^0$, alors les $\max(\mathcal{P})$ blocs de n éléments sont $|u^0 - \check{u}^0|$, ..., $|u^{v^0-1} - \check{u}^{v^0-1}|$, $\check{u}^{v^0}$ (sur n éléments), ..., $\check{u}^{\check{v}^0-1}$ (sur n éléments), suivis par des 0, si besoin.
      3. Le cas $v^0 > \check{v}^0$ est similaire, et donc omis
   3. Les p suivants sont $|v^1 - \check{v}^1|$, etc.

La fonction d peut se formaliser comme suit :

$$d(x, \check{x}) = d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) + d_{\mathbb{B}^{\mathsf{N}}}(e, \check{e}),$$

où :
— $d_{\mathbb{B}^{\mathsf{N}}}$ est la distance de Hamming,
— $\forall s = (u, v), \check{s} = (\check{u}, \check{v}) \in \mathcal{S}_{\mathsf{N},\mathcal{P}}$,

$$\begin{aligned} d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) \quad = \quad & \sum_{k=0}^{\infty} \frac{1}{10^{(k+1)p+kn\max(\mathcal{P})}} \Bigg(|v^k - \check{v}^k| \\ & + \left| \sum_{l=0}^{v^k-1} \frac{u^{\sum_{m=0}^{k-1} v^m + l}}{10^{(l+1)n}} - \sum_{l=0}^{\check{v}^k-1} \frac{\check{u}^{\sum_{m=0}^{k-1} \check{v}^m + l}}{10^{(l+1)n}} \right| \Bigg) \end{aligned}$$

Exemple. *On considère par exemple* $\mathsf{N} = 13$, $\mathcal{P} = \{1, 2, 11\}$ *(*p *vaut ainsi* 3*), et* $s = \begin{cases} u = \underline{6,}\ \underline{11,5}, \ldots \\ v = 1, 2, \ldots \end{cases}$ *avec* $\check{s} = \begin{cases} \check{u} = \underline{6,4}\ \underline{1}, \ldots \\ \check{v} = 2, 1, \ldots \end{cases}$ *. Ainsi*

$$d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s, \check{s}) = 0.01\ 0004000000000000000000\ 01\ 1005\ldots$$

En effet, les $p = 2$ *premiers éléments sont 01, c'est-à-dire* $|v^0 - \check{v}^0| = 1$*, et on utilise* p *éléments pour représenter cette différence (Comme* $\mathcal{P} = \{1, 2, 11\}$*, cette différence peut valoir 10). On prend alors le* $v^0 = 1$ *premier terme de* u*, chaque terme étant codé sur*

$n = 2$ éléments, soit 06. Comme on itère au plus $\max(\mathcal{P})$ fois, on complète cette valeur par des 0 de sorte que la chaîne obtenue ait $n \times \max(\mathcal{P}) = 22$ éléments, soit : 0600000000000000000000. De manière similaire, les $\check{v}^0 = 2$ premiers termes de $\check{u}$ sont représentés par 0604000000000000000000. La valeur absolue de leur différence est égale à 0004000000000000000000. Ces éléments sont concaténés avec 01. On peut construire alors le reste de la séquence.

On a la proposition suivante, qui est démontrée en annexe C.

Théorème [18] (Une distance dans $\mathcal{X}_{\mathsf{N},\mathcal{P}}$)**.** *$d$ est une distance sur $\mathcal{X}_{\mathsf{N},\mathcal{P}}$.*

5.3.3/ LE GRAPHE $\mathrm{GIU}_{\mathcal{P}}(f)$ ÉTENDANT $\mathrm{GIU}(f)$

A partir de $\mathcal{P} = \{p_1, p_2, \ldots, p_{\mathsf{p}}\}$, on définit le graphe orienté $\mathrm{GIU}_{\mathcal{P}}(f)$ de la manière suivante :
- les nœuds sont les 2^{N} configurations de $\mathbb{B}^{\mathsf{N}}$,
- il y a un arc libellé $u_0, \ldots, u_{p_i-1}$, $i \in [\![1, \mathsf{p}]\!]$ entre les nœuds x et y si et seulement si p_i est un élément de $\mathcal{P}$ (*i.e.*, on peut itérer p_i fois), et pour chaque k, $0 \leq k \leq p_i - 1$, on a u_k qui appartient à [N] et $y = F_{f_u,p_i}(x, (u_0, \ldots, u_{p_i-1}))$.

Il n'est pas difficile de constater que $\mathrm{GIU}_{\{1\}}(f)$ est $\mathrm{GIU}(f)$.

Exemple. *On reprend l'exemple où* $\mathsf{N} = 2$ *et* $h(x_1, x_2) = (\overline{x_1}, x_1\overline{x_2} + \overline{x_1}x_2)$ *déjà détaillé à la section 5.2.2.*

Le graphe $\mathrm{GIU}_{\{1\}}(h)$ *a déjà été donné à la figure 5.1(b). Les graphes* $\mathrm{GIU}_{\{2\}}(h)$, $\mathrm{GIU}_{\{3\}}(h)$ *et* $\mathrm{GIU}_{\{2,3\}}(h)$ *sont respectivement donnés aux figures 5.5(a), 5.5(b) et 5.5(c). Le premier (respectivement le second) illustre le comportement du générateur lorsque qu'on itère exactement 2 fois (resp. 3 fois) puis qu'on affiche le résultat. Le dernier donnerait le comportement d'un générateur qui s'autoriserait à itérer en interne systématiquement 2 ou 3 fois avant de retourner un résultat.*

5.3.4/ LE PRNG DE L'ALGORITHME 1 EST CHAOTIQUE SUR $\mathcal{X}_{\mathsf{N},\mathcal{P}}$

Le théorème suivant, similaire à ceux dans $\mathcal{X}_u$ et dans $\mathcal{X}_g$ est prouvé en annexe C.

Théorème [19] (Conditions pour la chaoticité de $G_{f_u,\mathcal{P}}$)**.** *La fonction $G_{f_u,\mathcal{P}}$ est chaotique sur $(\mathcal{X}_{\mathsf{N},\mathcal{P}}, d)$ si et seulement si le graphe d'itérations $\mathrm{GIU}_{\mathcal{P}}(f)$ est fortement connexe.*

5.4/ CONCLUSION

Ce chapitre a proposé un algorithme permettant de construire un PRNG chaotique à partir d'un PRNG existant. Pour ce faire, il est nécessaire et suffisant que la fonction f qui est itérée un nombre b de fois possède un $\mathrm{GIU}_{\{b\}}(f)$ fortement connexe et que sa matrice de Markov associée soit doublement stochastique. Le chapitre suivant montre comment construire une telle fonction.

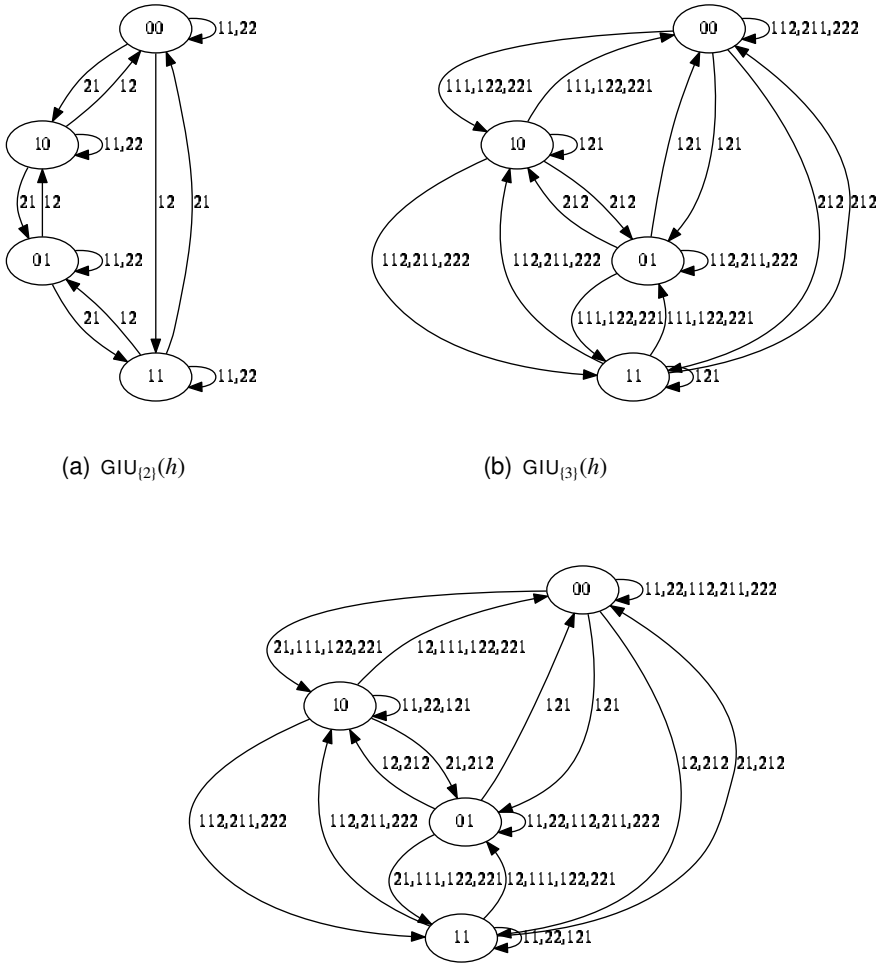


FIGURE 5.5 – Graphes d'itérations $\mathrm{GIU}_{\mathcal{P}}(h)$ pour $h(x_1, x_2) = (\overline{x_1}, x_1\overline{x_2} + \overline{x_1}x_2)$

6

LES GÉNÉRATEURS ISSUS DES CODES DE GRAY

On a vu dans le chapitre précédent que pour avoir un générateur à sortie uniforme, il est nécessaire que la matrice d'adjacence du graphe d'itération du système soit doublement stochastique. Nous présentons dans cette partie des méthodes effectives permettant de générer de telles matrices. La première est basée sur la programmation logique par contraintes (Section 6.1). Cependant celle-ci souffre de ne pas passer à l'échelle et ne fournit pas une solution en un temps raisonnable dès que la fonction à engendrer porte sur un grand nombre de bits. Une approche plus pragmatique consiste à supprimer un cycle hamiltonien dans le graphe d'itérations GIU($\neg$) (section 6.2). Pour obtenir plus rapidement une distribution uniforme, l'idéal serait de supprimer un cycle hamiltonien qui nierait autant de fois chaque bit. Cette forme de cycle est dite équilibré. La section 6.3 établit le lien avec les codes de Gray équilibrés, étudiés dans la littérature. La section suivante présente une démarche de génération automatique de code de Gray équilibré (section 6.4). La vitesse avec laquelle l'algorithme de PRNG converge en interne vers une distribution uniforme est étudiée théoriquement et pratiquement à la section 6.5. L'extension du travail aux itérations généralisées est présentée à la section 6.6. Finalement, des instances de PRNGs engendrés selon les méthodes détaillées dans ce chapitre sont présentées en section 6.7. Les sections 6.1 à 6.3 ont été publiées à [CHG+14a]. La section 6.5 est publiée dans [CCVHG16].

6.1/ PROGRAMMATION LOGIQUE PAR CONTRAINTES SUR DES DOMAINES FINIS

Tout d'abord, soit N le nombre d'éléments. Pour éviter d'avoir à gérer des fractions, on peut considérer que les matrices (d'incidence) à générer ont des lignes et des colonnes dont les sommes valent N à chaque fois. On cherche ainsi toutes les matrices M de taille $2^N \times 2^N$ telles que

1. M_{ij} vaut 0 si j n'est pas un voisin de i, *i.e.*, il n'y a pas d'arc de i à j dans le graphe GIU(f) ;
2. $0 \leq M_{ii} \leq$ N : le nombre de boucles autour de chaque configuration i est inférieur à N ;
3. pour $j \neq i$, $0 \leq M_{ij} \leq 1$: on construit l'arc de i à j si et seulement si M_{ij} vaut 1 (et 0

sinon) ;

4. pour chaque indice de ligne i, $1 \leq i \leq 2^{\mathrm{N}}$, $\mathrm{N} = \sum_{1 \leq j \leq 2^{\mathrm{N}}} M_{ij}$: la matrice est stochastique à droite ;
5. pour chaque indice de colonne j, $1 \leq j \leq 2^{\mathrm{N}}$, $\mathrm{N} = \sum_{1 \leq i \leq 2^{\mathrm{N}}} M_{ij}$: la matrice est stochastique à gauche ;
6. Tous les éléments de la somme $\sum_{1 \leq k \leq 2^{\mathrm{N}}} M^k$ sont strictement positifs, *i.e.*, le graphe $\mathrm{GIU}(f)$ est fortement connexe ;

Ce problème s'exprime sur des domaines finis entiers avec des opérateurs arithmétiques simples (sommes et produits). Il pourrait théoriquement être traité par des démarches de programmation logique par contrainte sur des domaines finis (comme en PROLOG). L'algorithme donné en Figure 6.1 est en effet le cœur du programme PROLOG qui pourrait être utilisé pour instancier toutes les solutions, ici pour $\mathrm{N} = 2$. Dans ce code, $\mathtt{mmult}(X, Y, R)$ et $\mathtt{summ}(X, Y, R)$ valent True si et seulement si R est le produit matriciel (ou la somme matricielle) entre X et Y respectivement. Il n'est pas difficile d'adapter ce code à n'importe quelle valeur entière naturelle N.

```
bistoc(X):-
  M=[[M0_0, M0_1, 0, M0_3], [M1_0, M1_1, 0, M1_3],
     [M2_0, 0, M2_2, M2_3], [0, M3_1, M3_2, M3_3]],
  [M0_0, M1_1, M2_2, M3_3] ins 0..2,
  [M0_1, M0_3, M1_0, M1_3, M2_0, M2_3, M3_1, M3_2]
     ins 0..1,
  M0_0+ M0_1+ M0_2 #=2, M1_0+ M1_1+ M1_3 #=2,
  M2_0+ M2_2+ M2_3 #=2, M3_1+ M3_2+ M3_3 #=2,
  M0_0+ M1_0+ M2_0 #=2, M0_1+ M1_1+ M3_1 #=2,
  M0_2+ M2_2+ M3_2 #=2, M1_3+ M2_3+ M3_3 #=2,
  mmult(M,M,M2),
  mmult(M,M2,M3),
  mmult(M,M3,M4),
  summ(M,M2,S2),
  summ(S2,M3,S3),
  summ(S3,M4,S4),
  allpositive(S4).
```

FIGURE 6.1 – Code PROLOG permettant de trouver toutes les matrices DSSC pour $n = 2$

Enfin, on définit la relation $\mathcal{R}$, qui est établie pour les deux fonctions f et g si leurs graphes respectifs $\mathrm{GIU}(f)$ et $\mathrm{GIU}(g)$ sont isomorphes. C'est évidemment une relation d'équivalence.

Exécutée sur un ordinateur personnel, PROLOG trouve en moins d'une seconde les 49 solutions pour $n = 2$, dont 2 seulement ne sont pas équivalentes, en moins d'une minute les 27642 solutions dont seulement 111 ne sont pas équivalentes pour $n = 3$. Cependant, l'approche ne permet pas d'engendrer toutes les solutions pour $n = 4$. Cette approche, basée sur une démarche de type *générer, tester* ne peut pas être retenue pour n de grande taille, même en s'appuyant sur l'efficience de l'algorithme de backtrack natif de PROLOG.

Cependant, pour des petites valeurs de n, nous avons comparé les fonctions non équivalentes selon leur proportion à engendrer des temps de mélange petits (cf. équation (5.2)).

Exemple. *Le tableau 6.1 fournit les 5 fonctions booléennes qui ont les temps de mélange les plus petits pour* $\varepsilon = 10^{-5}$.

Name	Image	MT
f^a	$(x_2 \oplus x_3, x_1 \oplus \overline{x_3}, \overline{x_3})$	16
f^*	$(x_2 \oplus x_3, \overline{x_1}.\overline{x_3} + x_1\overline{x_2}, \overline{x_1}.\overline{x_3} + x_1x_2)$	17
f^b	$(\overline{x_1}(x_2 + x_3) + x_2x_3, \overline{x_1}(\overline{x_2} + \overline{x_3}) + \overline{x_2}.\overline{x_3},$ $\overline{x_3}(\overline{x_1} + x_2) + \overline{x_1}x_2)$	26
f^c	$(\overline{x_1}(x_2 + x_3) + x_2x_3, \overline{x_1}(\overline{x_2} + \overline{x_3}) + \overline{x_2}.\overline{x_3},$ $\overline{x_3}(\overline{x_1} + x_2) + \overline{x_1}x_2)$	29
f^d	$(x_1 \oplus x_2, x_3(\overline{x_1} + \overline{x_2}), \overline{x_3})$	30

TABLE 6.1 – Les 5 fonctions booléennes $n = 3$ aux temps de mélange les plus faibles.

Une analyse syntaxique de ces fonctions ne permet pas, a priori, de déduire des règles permettant de construire de nouvelles fonctions dont le temps de mélange serait faible. Cependant, le graphe GIU(f^*) (donné à la Figure 6.4(a)) est le 3-cube dans lequel le cycle $000, 100, 101, 001, 011, 111, 110, 010, 000$ a été enlevé. Dans cette figure, le graphe GIU(f) est en continu tandis que le cycle est en pointillés. Ce cycle qui visite chaque nœud exactement une fois est un *cycle hamiltonien*. La matrice de Markov correspondante est donnée à la figure 6.4(b). On s'intéresse par la suite à la génération de ce genre de cycles.

6.2/ SUPPRESSION DES CYCLES HAMILTONIENS

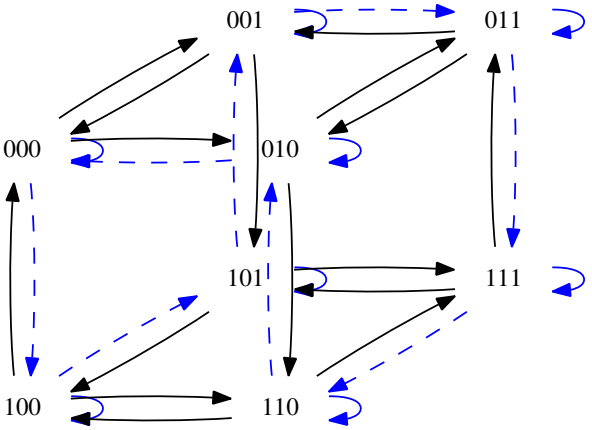
Dans un premier temps, nous montrons que la suppression d'un cycle hamiltonien produit bien des matrices doublement stochastiques. Nous établissons ensuite le lien avec les codes de Gray équilibrés.

Nous donnons deux résultats complémentaires, reliant la suppression d'un cycle hamiltonien du n-cube, les matrices doublement stochastiques et la forte connexité du graphe d'itérations.

Théorème 20. *La suppression d'un cycle hamiltonien dans une matrice de Markov M, issue du n-cube, produit une matrice doublement stochastique.*

Démonstration. Un cycle hamiltonien passe par chaque nœud une et une seule fois. Pour chaque nœud v dans le n-cube C_1, une arête entrante (o, v) et une arête sortante (v, e) sont ainsi enlevées. Considérons un autre n-cube C_2 auquel on ajoute les arêtes pour le rendre complet. La matrice de Markov M correspondante est remplie de $\frac{1}{2^n}$ et est doublement stochastique. Comme on enlève exactement une arête sortante (v, e) à chaque nœud v dans C_1, on enlève ainsi dans C_2 les 2^{n-1} arêtes issues de v et relatives aux parties de $\{1, \ldots, n\}$ qui contiennent e. Cela revient à annuler dans la matrice M les 2^{n-1} coefficients correspondants et ajouter $1/2$ à l'élément M_{vv}. La matrice M reste stochastique. On effectue un raisonnement similaire pour les arêtes entrantes : comme on enlève exactement une arête entrante (o, v) pour chaque nœud v dans C_1, dans C_2, ce sont exactement 2^{n-1} arêtes menant à v qui sont enlevées. Dans M les 2^{n-1} coefficients correspondants sont mis à 0 et M_{vv} vaut alors $\frac{2^{n-1}+1}{2}$. M est donc doublement stochastique. □

Théorème 21. *Le graphe d'itérations issu du n-cube, auquel un cycle hamiltonien est enlevé, est fortement connexe.*



(a) Graphe GIU(f^*).

$$M = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

(b) Matrice de Markov associée à GIU(f^*)

FIGURE 6.2 – Représentations de $f^*(x_1, x_2, x_3) = (x_2 \oplus x_3, \overline{x_1}.\overline{x_3} + x_1\overline{x_2}, \overline{x_1}.\overline{x_3} + x_1x_2)$.

Démonstration. On considère les deux n-cubes C_1 et C_2 définis dans la preuve du théorème 20. Dans C_1 on considère le cycle inverse r du cycle hamiltonien c. Aucun arc n'appartient à la fois à r et à c : en effet, sinon c contiendrait un nœud deux fois. Ainsi aucune arête de r n'est enlevée dans C_1. Le cycle r est évidemment un cycle hamiltonien et contient tous les nœuds. Tous les nœuds de C_1 dans lesquels c a été enlevé sont accessibles depuis n'importe quel nœud. Le graphe des itérations GIU qui étend le précédent graphe est ainsi fortement connexe.

□

Générer un cycle hamiltonien dans le n-cube revient à trouver un *code de Gray cyclique*. On rappelle qu'un code de Gray est une séquence de mots binaires de taille fixe (N), dont les éléments successifs ne différent que par un seul bit. Un code de Gray est *cyclique* si le premier élément et le dernier ne différent que par un seul bit.

6.3/ LIEN AVEC LES CODES DE GRAY CYCLIQUES (TOTALEMENT) ÉQUILIBRÉS

Un minorant du nombre de codes de Gray $\left(\left(\frac{n*\log 2}{e \log\log n} \times (1 - o(1))\right)^{2^n}\right)$, donnée dans [FS09], indique une explosion combinatoire pour notre recherche. Afin de contourner cette difficulté, nous nous restreignons aux codes induisant un graphe d'itérations GIU(f) *uniforme*. Cette uniformité se traduit par des nombres d'arcs équilibrés entre les *dimensions* du graphe, la dimension i correspondant aux seules variations du bit i (parmi les n bits au total). Cette approche revient à chercher des codes de Gray cycliques *équilibrés*.

On formalise un code de Gray équilibré comme suit. Soit $L = w_1, w_2, \ldots, w_{2^n}$ la séquence d'un code de Gray cyclique à n bits. Soit $S = s_1, s_2, \ldots, s_{2^n}$ la séquence des transitions où s_i, $1 \leq i \leq 2^n$ est l'indice du seul bit qui varie entre les mots w_i et w_{i+1}. Enfin, soit TC_n : $\{1, \ldots, n\} \rightarrow \{0, \ldots, 2^n\}$ la fonction qui au paramètre i associe le *nombre de transitions* présentes dans la séquence L pour le bit i, c'est-à-dire le nombre d'occurrences de i

dans S.

Le code L est **équilibré** si $\forall i, j \in \{1, ..., n\}$, $|TC_n(i) - TC_n(j)| \leq 2$. Il est **totalement équilibré** si $\forall i \in \{1, ..., n\}$, $TC_n(i) = \frac{2^n}{n}$.

On peut donc déjà déduire qu'il ne peut exister des codes de Gray totalement équilibrés que pour les systèmes ayant un nombre d'éléments $n = 2^k, k > 0$. De plus, comme dans tout code de Gray cyclique, $TC_n(i)$ est pair $\forall i \in \{1, ..., n\}$, alors les systèmes ayant un nombre d'éléments différent de 2^k, ne peuvent avoir que des codes de Gray équilibrés avec $TC_n(i) = \lfloor \frac{2^n}{n} \rfloor$ ou $TC_n(i) = \lceil \frac{2^n}{n} \rceil, \forall i \in \{1, ..., n\}$ et vérifiant $\sum_{i=1}^{n} TC_n(i) = 2^n$.

Exemple. *Soit* $L^* = 000, 100, 101, 001, 011, 111, 110, 010$ *le code de Gray correspondant au cycle hamiltonien enlevé de* f^**. Sa séquence des transitions est* $S = 3, 1, 3, 2, 3, 1, 3, 2$ *et les nombres de transitions sont* $TC_3(1) = TC_3(2) = 2$ *et* $TC_3(3) = 4$*. Un tel code est équilibré.*

Si l'on considère maintenant L^4 *= 0000, 0010, 0110, 1110, 1111, 0111, 0011, 0001, 0101, 0100, 1100, 1101, 1001, 1011, 1010, 1000, un code de Gray cyclique. En construisant la séquence* S *= 2,3,4,1,4,3,2,3,1,4,1,3,2,1,2,4, on constate que* $\forall i \in \{1, .., 4\}$, $TC_4(i) = 4$ *et donc que ce code est totalement équilibré.*

6.4/ GÉNÉRATION DE CODES DE GRAY ÉQUILIBRÉS PAR INDUCTION

De nombreuses approches ont été développées pour résoudre le problème de construire un code de Gray dans un N-cube [RC81, BS96, ZS04], selon les propriétés que doit vérifier ce code.

Dans les travaux [RC81], les auteurs proposent une approche inductive de construction de code de Gray équilibrés (on passe du $N - 2$ à N) pour peu que l'utilisateur fournisse une sous-séquence possédant certaines propriétés à chaque pas inductif. Ce travail a été renforcé dans [BS96] où les auteurs donnent une manière explicite de construire une telle sous-séquence. Enfin, les auteurs de [ZS04] présentent une extension de l'algorithme de *Robinson-Cohn*. La présentation rigoureuse de cette extension leur permet principalement de prouver que si N est une puissance de 2, le code de Gray équilibré engendré par l'extension est toujours totalement équilibré et que S_N est la séquence de transition d'un code de Gray de N bits si S_{N-2} l'est aussi. Cependant les auteurs ne prouvent pas que leur approche fournit systématiquement un code de Gray (totalement) équilibré. Cette section montre que ceci est vrai en rappelant tout d'abord l'extension de l'algorithme de *Robinson-Cohn* pour un code de Gray avec $N - 2$ bits défini à partir de la séquence S_{N-2}.

1. Soit l un entier positif pair. Trouver des sous-séquences $u_1, u_2, \ldots, u_{l-2}, v$ (possiblement vides) de S_{N-2} telles que S_{N-2} est la concaténation de

$$s_{i_1}, u_0, s_{i_2}, u_1, s_{i_3}, u_2, \ldots, s_{i_l-1}, u_{l-2}, s_{i_l}, v$$

   où $i_1 = 1$, $i_2 = 2$, et $u_0 = \emptyset$ (la séquence vide).

2. Remplacer dans S_{N-2} les séquences $u_0, u_1, u_2, \ldots, u_{l-2}$ par $N - 1, u'(u_1, N - 1, N), u'(u_2, N, N - 1), u'(u_3, N - 1, N), \ldots, u'(u_{l-2}, N, N - 1)$ respectivement, où $u'(u, x, y)$

est la séquence u, x, u^R, y, u telle que u^R est u, mais dans l'ordre inverse. La séquence obtenue est ensuite notée U.

3. Construire les séquences $V = v^R, \mathrm{N}, v$, $W = \mathrm{N}-1, S_{\mathrm{N}-2}, \mathrm{N}$. Soit alors W' définie comme étant égale à W sauf pour les deux premiers éléments qui ont été intervertis.
4. La séquence de transition S_{N} est la concaténation U^R, V, W'.

L'étape (1.) n'est pas constructive : il n'est pas précisé comment sélectionner des sous-séquences qui assurent que le code obtenu est équilibré. Le théorème suivant montre que c'est possible et sa preuve, donnée en annexe C, explique comment le faire.

Théorème [22] (Existence d'un code de Gray équilibré)**.** *Soit* N *dans* $\mathbb{N}^*$*, et* a_{N} *défini par* $a_{\mathrm{N}} = 2\left\lfloor \frac{2^{\mathrm{N}}}{2\mathrm{N}} \right\rfloor$*. Il existe une séquence* l *dans l'étape (1.) de l'extension de l'algorithme de* Robinson-Cohn *telle que les nombres de transitions* $TC_{\mathrm{N}}(i)$ *valent tous* a_{N} *ou* $a_{\mathrm{N}} + 2$ *pour chaque* i*,* $1 \leq i \leq \mathrm{N}$.

Ces fonctions étant générées, on s'intéresse à étudier à quelle vitesse un générateur les embarquant converge vers la distribution uniforme. C'est l'objectif de la section suivante.

6.5/ QUANTIFIER L'ÉCART PAR RAPPORT À LA DISTRIBUTION UNIFORME

On considère ici une fonction construite comme à la section précédente. On s'intéresse ici à étudier de manière théorique les itérations définies à l'équation (3.2) pour une stratégie donnée. Tout d'abord, celles-ci peuvent être interprétées comme une marche le long d'un graphe d'itérations GIU(f) tel que le choix de tel ou tel arc est donné par la stratégie. On remarque que ce graphe d'itérations est toujours un sous graphe du N-cube augmenté des boucles sur chaque sommet, *i.e.*, les arcs (v, v) pour chaque $v \in \mathbb{B}^{\mathrm{N}}$. Ainsi, le travail ci-dessous répond à la question de définir la longueur du chemin minimum dans ce graphe pour obtenir une distribution uniforme. Ceci se base sur la théorie des chaînes de Markov. Pour une référence générale à ce sujet on pourra se référer au livre [LPW06], particulièrement au chapitre sur les temps d'arrêt.

Exemple. *On considère par exemple le graphe* GIU(f) *donné à la* FIGURE 6.4(A)*. et la fonction de probabilités* p *définie sur l'ensemble des arcs comme suit :*

$$p(e) \begin{cases} = \frac{1}{2} + \frac{1}{6} \; si \; e = (v, v) \; avec \; v \in \mathbb{B}^3, \\ = \frac{1}{6} \; sinon. \end{cases}$$

La matrice P *de la chaîne de Markov associée à* f^* *est*

$$P = \frac{1}{6} \begin{pmatrix} 4 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 4 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 4 \end{pmatrix}$$

On remarque que dans cette marche on reste sur place avec une probabilité égale à $\frac{1}{2}+\frac{1}{2N}$ et l'on passe d'un sommet à son voisin lorsque c'est possible avec une probabilité $\frac{1}{2N}$. Les probabilités usuelles que l'on appliquerait aux transitions de l'algorithme 1 seraient quant à elles uniformément égales à $\frac{1}{N}$. Cette manière paresseuse d'itérer (puisqu'on reste plus souvent sur place) n'est donc pas équivalente à celle issue de l'algorithme.

Cependant, l'étude théorique de référence [LPW06] considère cette marche comme cadre. S'inspirant de celle-ci, le travail suivant se replace donc dans ce cadre théorique.

Tout d'abord, soit π et μ deux distributions sur $\mathbb{B}^N$. La distance de « totale variation » entre π et μ est notée $||\pi - \mu||_{TV}$ et est définie par

$$||\pi - \mu||_{TV} = \max_{A \subset \mathbb{B}^N} |\pi(A) - \mu(A)|.$$

On sait que

$$||\pi - \mu||_{TV} = \frac{1}{2} \sum_{X \in \mathbb{B}^N} |\pi(X) - \mu(X)|.$$

De plus, si ν est une distribution sur $\mathbb{B}^N$, on a

$$||\pi - \mu||_{TV} \leq ||\pi - \nu||_{TV} + ||\nu - \mu||_{TV}.$$

Soit P une matrice d'une chaîne de Markov sur $\mathbb{B}^N$. $P(X, \cdot)$ est la distribution induite par la $X^{\text{ème}}$ colonne de P. Si la chaîne de Markov induite par P a une distribution stationnaire π, on définit alors

$$d(t) = \max_{X \in \mathbb{B}^N} ||P^t(X, \cdot) - \pi||_{TV}$$

et

$$t_{mix}(\varepsilon) = \min\{t \mid d(t) \leq \varepsilon\}.$$

Intuitivement, $t_{mix}(\varepsilon)$ est le nombre d'itérations nécessaire pour être proche de la distribution stationnaire à ε près, peu importe la configuration de départ. On a le théorème suivant démontré en annexe C.

Théorème [23] (Temps de mixage sans chemin hamiltonien)**.** *On considère un* N*-cube dans lequel un chemin hamiltonien a été supprimé et la fonction de probabilités p définie sur l'ensemble des arcs comme suit :*

$$p(e) \begin{cases} = \frac{1}{2} + \frac{1}{2N} \text{ si } e = (v, v) \text{ avec } v \in \mathbb{B}^N, \\ = \frac{1}{2N} \text{ sinon.} \end{cases}$$

La chaîne de Markov associée converge vers la distribution uniforme et

$$\forall \varepsilon > 0,\ t_{mix}(\varepsilon) \leq x \leq \lceil \log_2(\varepsilon^{-1})(32N^2 + 16N \ln(N + 1)) \rceil$$

Sans entrer dans les détails de la preuve, on remarque aussi que le calcul de ce majorant impose uniquement que pour chaque nœud du N-cube un arc entrant et un arc sortant sont supprimés. Le fait qu'on enlève un cycle hamiltonien et que ce dernier soit équilibré n'est pas pris en compte. En intégrant cette contrainte, ce majorant pourrait être réduit.

En effet, le temps de mixage est en $\Theta(N \ln N)$ lors d'une marche aléatoire classique paresseuse dans le N-cube. On peut ainsi conjecturer que cet ordre de grandeur reste le même dans le contexte du N-cube privé d'un chemin hamiltonien.

On peut évaluer ceci pratiquement : pour une fonction $f : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ et une graine initiale x^0, le code donné à l'algorithme 2 retourne le nombre d'itérations suffisant tel que tous les éléments $\ell \in [\![1, \mathsf{N}]\!]$ sont équitables. Il permet de déduire une approximation de $E[\tau_{\text{stop}}]$ en l'instanciant un grand nombre de fois : pour chaque nombre N, $3 \leq \mathsf{N} \leq 16$, 10 fonctions ont été générées comme dans ce chapitre. Pour chacune d'elle, le calcul d'une approximation de $E[\tau_{\text{stop}}]$ est exécuté 10000 fois avec une graine aléatoire. La Figure 6.3 résume ces résultats. Dans celle-ci, un cercle représente une approximation de $E[\tau_{\text{stop}}]$ pour un N donné tandis que la courbe est une représentation de la fonction $x \mapsto 2x \ln(2x + 8)$. On constate que l'approximation de $E[\tau_{\text{stop}}]$ est largement inférieure au majorant quadratique donné au théorème 29 et que la conjecture donnée au paragraphe précédent est sensée.

Input : a function f, an initial configuration x^0 (N bits)
Output : a number of iterations *nbit*
$nbit \leftarrow 0$;
$x \leftarrow x^0$;
$fair \leftarrow \emptyset$;
while $|fair| < \mathsf{N}$ **do**
  $s \leftarrow Random(\mathsf{N})$;
  $image \leftarrow f(x)$;
  if $Random(1) \neq 0$ *and* $x[s] \neq image[s]$ **then**
    $fair \leftarrow fair \cup \{s\}$;
    $x[s] \leftarrow image[s]$;
  end
  $nbit \leftarrow nbit + 1$;
end
return *nbit*;

Algorithme 2 : Pseudo-code pour évaluer le temps d'arrêt

6.6/ ET LES ITÉRATIONS GÉNÉRALISÉES ?

Le chapitre précédent a présenté un algorithme de PRNG construit à partir d'itérations unaires. On pourrait penser que cet algorithme est peu efficace puisqu'il dispose d'une fonction f de $\mathbb{B}^n$ dans lui même mais il ne modifie à chaque itération qu'un seul élément de $[n]$. On pourrait penser à un algorithme basé sur les itérations généralisées, c'est-à-dire qui modifierait une partie des éléments de $[n]$ à chaque itération. C'est l'algorithme 3 donné ci-après.

Par rapport à l'algorithme 1 seule la ligne $s \leftarrow Set(Random(2^n))$ est différente. Dans celle-ci la fonction $Set : \{1, \ldots, 2^n\} \to \mathcal{P}(\{1, \ldots n\})$ retourne l'ensemble dont la fonction caractéristique serait représentée par le nombre donné en argument. Par exemple, pour $n = 3$, l'ensemble $Set(6)$ vaudrait $\{3, 2\}$. On remarque aussi que l'argument de la fonction *Random* passe de n à 2^n.

On a le théorème suivant qui étend le théorème 17 aux itérations généralisées.

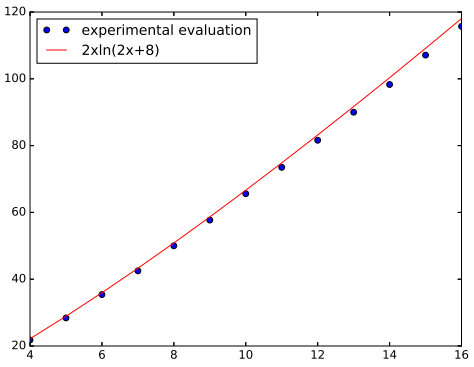


FIGURE 6.3 – Interpolation du temps d'arrêt

Théorème 24. *Soit* $f : \mathbb{B}^n \to \mathbb{B}^n$, GIG($f$) *son graphe des itérations généralisées,* $\check{M}$ *la matrice d'adjacence correspondante à ce graphe et* M *une matrice* $2^n \times 2^n$ *définie par* $M = \frac{1}{2^n}\check{M}$*. Si* GIG($f$) *est fortement connexe, alors la sortie du générateur de nombres pseudo-aléatoires détaillé par l'algorithme 3 suit une loi qui tend vers la distribution uniforme si et seulement si* M *est une matrice doublement stochastique.*

La preuve de ce théorème est la même que celle du théorème 17. Elle n'est donc pas rappelée.

Exemple. *On reprend l'exemple donné à la section 6.1. On considère le cycle hamiltonien défini par la séquence* $000, 100, 101, 001, 011, 111, 110, 010, 000$*. En supprimant celui-ci dans le* 3*-cube, cela engendre la fonction* f^* *définie par*

$$f^*(x_1, x_2, x_3) = (x_2 \oplus x_3, \overline{x_1}.\overline{x_3} + x_1\overline{x_2}, \overline{x_1}.\overline{x_3} + x_1x_2).$$

Le graphe GIG(f^*) *est représenté à la Figure 6.4(a). La matrice de Markov* M *correspondante est donnée à la figure 6.4(b).*

Le tableau 6.2 reprend une synthèse de fonctions qui ont été générées selon la méthode détaillée à la section 6.2. Pour chaque nombre $n = 3, 4, 5$ et 6, tous les cycles hamiltoniens non isomorphes ont été générés. Pour les valeur de $n = 7$ et 8, seuls 10^5 cycles ont été évalués. Parmi toutes les fonctions obtenues en enlevant du n-cube ces cycles, n'ont été retenues que celles qui minimisaient le temps de mélange relatif à une valeur de ϵ fixée à 10^{-8} et pour un mode donné. Ce nombre d'itérations (*i.e.*, ce temps de mélange) est stocké dans la troisième colonne sous la variable b. La variable b' reprend le temps de mélange pour l'algorithme 1. On note que pour un nombre n de bits fixé et un mode donné d'itérations, il peut y avoir plusieurs fonctions minimisant ce temps de mélange. De plus, comme ce temps de mélange est construit à partir de la matrice de Markov et que celle-ci dépend du mode, une fonction peut être optimale pour un mode et ne pas l'être pour l'autre (c.f. pour $n = 5$).

Un second résultat est que ce nouvel algorithme réduit grandement le nombre d'itérations suffisant pour obtenir une faible déviation par rapport à une distribution uniforme. On constate de plus que ce nombre décroît avec le nombre d'éléments alors qu'il augmente dans l'approche initiale où l'on marche.

Input : une fonction f, un nombre d'itérations b, une configuration initiale x^0 (n bits)
Output : une configuration x (n bits)
$x \leftarrow x^0$;
$k \leftarrow b$;
for $i = 1, \ldots, k$ **do**
  $s \leftarrow \mathit{Set}(\mathit{Random}(2^n))$;
  $x \leftarrow F_{f_g}(s, x)$;
end
return x;

Algorithme 3 : PRNG basé sur les itérations généralisées.

Cela s'explique assez simplement. Depuis une configuration initiale, le nombre de configurations qu'on ne peut pas atteindre en une itération est de :

— $2^n - n - 1$ en unaire. Ceci représente un rapport de $\frac{2^n - n - 1}{2^n} = 1 - \frac{n-1}{2^n}$ de toutes les configurations ; plus n est grand, plus ce nombre est proche de 1, et plus grand devient le nombre d'itérations nécessaires pour atteinte une déviation faible ;
— $2^n - 2^{n-1}$ dans le cas généralisé, soit la moitié de toutes les configurations quel que soit n ; seul 1 bit reste constant tandis que tous les autres peuvent changer. Plus n grandit, plus la proportion de bits constants diminue.

Cependant, dans le cas généralisé, chaque itération a une complexité plus élevée puisqu'il est nécessaire d'invoquer un générateur produisant un nombre pseudo-aléatoire dans $[2^n]$ tandis qu'il suffit que celui-ci soit dans $[n]$ dans le cas unaire. Pour comparer les deux approches, on considère que le générateur aléatoire embarqué est binaire, *i.e.* ne génère qu'un bit (0 ou 1).

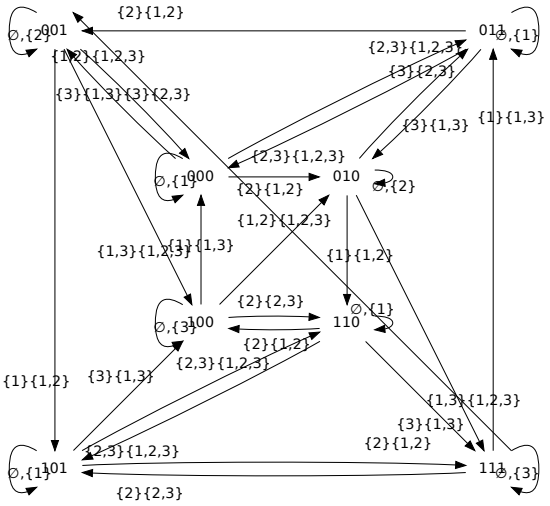
Dans le cas généralisé, si l'on effectue b itérations, à chacune d'elles, la stratégie génère un nombre entre 1 et 2^n. Elle fait donc n appels à ce générateur. On fait donc au total $b * n$ appels pour n bits et donc b appels pour 1 bit généré en moyenne. Dans le cas unaire, si l'on effectue b' itérations, à chacune d'elle, la stratégie génère un nombre entre 1 et n. Elle fait donc $\ln(n)/\ln(2)$ appels à ce générateur binaire en moyenne. La démarche fait donc au total $b' * \ln(n)/\ln(2)$ appels pour n bits et donc $b' * \ln(n)/(n * \ln(2))$ appels pour 1 bit généré en moyenne. Le tableau 6.3 donne des instances de ces valeurs pour $n \in \{4, 5, 6, 7, 8\}$ et les fonctions données au tableau 6.2. On constate que le nombre d'appels par bit généré décroît avec n dans le cas des itérations généralisées et est toujours plus faible que celui des itérations unaires.

6.7/ TESTS STATISTIQUES

La qualité des séquences aléatoires produites par le générateur des itérations unaires ainsi que celles issues des itérations généralisées a été évaluée à travers la suite de tests statistiques développée par le *National Institute of Standards and Technology* (NIST). En interne, c'est l'implantation de l'algorithme de Mersenne Twister qui permet de générer la stratégie aléatoire.

Pour les 15 tests, le seuil α est fixé à 1% : une valeur qui est plus grande que 1% signifie que la chaîne est considérée comme aléatoire avec une confiance de 99%.

Les tableaux 6.4 donnent une vision synthétique de ces expérimentations. Nous avons



(a) Graphe GIG(f^*)

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

(b) Matrice de Markov associée au GIG(f^*)

FIGURE 6.4 – Représentations de $f^*(x_1, x_2, x_3) = (x_2 \oplus x_3, \overline{x_1}.\overline{x_3} + x_1\overline{x_2}, \overline{x_1}.\overline{x_3} + x_1x_2)$.

évalué les fonctions préfixées par f (respectivement g) avec les générateurs issus des itérations généralisées (resp. unaires). Quelle que soit la méthode utilisée, on constate que chacun des générateurs passe avec succès le test de NIST.

Interpréter ces résultats en concluant que ces générateurs sont tous équivalents serait erroné : la meilleure des méthodes basées sur le mode des itérations généralisées (pour $n = 8$ par exemple) est au moins deux fois plus rapide que la meilleure de celles qui sont basées sur les itérations unaires.

6.8/ CONCLUSION

Ce chapitre a montré comment construire un PRNG chaotique, notamment à partir de codes de Gray équilibrés. Une méthode complètement automatique de construction de ce type de codes a été présentée étendant les méthodes existantes. Dans le cas des itérations unaires, l'algorithme qui en découle a un temps de mélange qui a un majorant quadratique de convergence vers la distribution uniforme. Pratiquement, ce temps de mélange se rapproche de $N \ln N$. Les expérimentations au travers de la batterie de test de NIST ont montré que toutes les propriétés statistiques sont obtenues pour N $= 4, 5, 6, 7, 8$.

n	fonction	$f(x)$, $f(x)$ pour $x \in [0, 1, 2, \ldots, 2^n - 1]$	b	b'
4	f^{*4}	[13,10,9,14,3,11,1,12,15,4,7,5,2,6,0,8]	**17**	**38**
5	f^{*5}	[29, 22, 25, 30, 19, 27, 24, 16, 21, 6, 5, 28, 23, 26, 1, 17, 31, 12, 15, 8, 10, 14, 13, 9, 3, 2, 7, 20, 11, 18, 0, 4]	**13**	48
	g^{*5}	[29, 22, 21, 30, 19, 27, 24, 28, 7, 20, 5, 4, 23, 26, 25, 17, 31, 12, 15, 8, 10, 14, 13, 9, 3, 2, 1, 6, 11, 18, 0, 16	15	**47**
6	f^{*6}	[55, 60, 45, 56, 58, 42, 61, 40, 53, 50, 52, 54, 59, 34, 33, 49, 39, 62, 47, 46, 11, 43, 57, 8, 37, 6, 36, 4, 51, 38, 1, 48, 63, 26, 25, 30, 19, 27, 17, 28, 31, 20, 23, 21, 18, 22, 16, 24, 13, 12, 29, 44, 10, 14, 41, 0, 15, 2, 7, 5, 35, 3, 9, 32]	**11**	55
	g^{*6}	[55, 60, 45, 44, 43, 62, 61, 48, 53, 50, 52, 36, 59, 51, 33, 49, 15, 14, 47, 46, 35, 58, 57, 56, 7, 54, 39, 37, 3, 38, 1, 40, 63, 26, 25, 30, 19, 27, 17, 28, 31, 20, 23, 21, 18, 22, 16, 24, 13, 12, 29, 8, 10, 42, 41, 0, 5, 2, 4, 6, 11, 34, 9, 32]	12	**54**
7	f^{*7}	[111, 94, 93, 116, 122, 114, 125, 88, 115, 126, 85, 84, 123, 98, 81, 120, 109, 78, 105, 110, 99, 107, 104, 108, 101, 118, 117, 96, 103, 66, 113, 64, 79, 86, 95, 124, 83, 91, 121, 24, 119, 22, 69, 20, 87, 18, 17, 112, 77, 76, 73, 12, 74, 106, 72, 8, 7, 102, 71, 100, 75, 82, 97, 0, 127, 54, 57, 62, 51, 59, 56, 48, 53, 38, 37, 60, 55, 58, 33, 49, 63, 44, 47, 40, 42, 46, 45, 41, 35, 34, 39, 52, 43, 50, 32, 36, 29, 28, 61, 92, 26, 90, 89, 25, 19, 30, 23, 4, 27, 2, 16, 80, 31, 10, 15, 14, 3, 11, 13, 9, 5, 70, 21, 68, 67, 6, 65, 1]	**10**	**63**
8	f^{*8}	[223, 190, 249, 254, 187, 251, 233, 232, 183, 230, 247, 180, 227, 178, 240, 248, 237, 236, 253, 172, 203, 170, 201, 168, 229, 166, 165, 244, 163, 242, 241, 192, 215, 220, 205, 216, 218, 222, 221, 208, 213, 210, 212, 214, 219, 211, 217, 209, 239, 202, 207, 140, 139, 234, 193, 204, 135, 196, 199, 132, 194, 130, 225, 200, 159, 62, 185, 252, 59, 250, 169, 56, 191, 246, 245, 52, 243, 50, 176, 48, 173, 238, 189, 44, 235, 42, 137, 184, 231, 38, 37, 228, 35, 226, 177, 224, 151, 156, 141, 152, 154, 158, 157, 144, 149, 146, 148, 150, 155, 147, 153, 145, 175, 206, 143, 12, 11, 142, 129, 128, 7, 198, 197, 4, 195, 2, 161, 160, 255, 124, 109, 108, 122, 126, 125, 112, 117, 114, 116, 100, 123, 98, 97, 113, 79, 106, 111, 110, 99, 74, 121, 120, 71, 118, 103, 101, 115, 66, 65, 104, 127, 90, 89, 94, 83, 91, 81, 92, 95, 84, 87, 85, 82, 86, 80, 88, 77, 76, 93, 72, 107, 78, 105, 64, 69, 102, 68, 70, 75, 67, 73, 96, 55, 58, 45, 188, 51, 186, 61, 40, 119, 182, 181, 53, 179, 54, 33, 49, 15, 174, 47, 60, 171, 46, 57, 32, 167, 6, 36, 164, 43, 162, 1, 0, 63, 26, 25, 30, 19, 27, 17, 28, 31, 20, 23, 21, 18, 22, 16, 24, 13, 10, 29, 14, 3, 138, 41, 136, 39, 134, 133, 5, 131, 34, 9, 8]	9	71

TABLE 6.2 – Fonctions avec matrices DSCC et le plus faible temps de mélange

Itérations	4	5	6	7	8
Unaires	19.0	22.3	23.7	25.3	27.0
Généralisées	17	13	11	10	9

TABLE 6.3 – Nombre moyen d'appels à un générateur binaire par bit généré

Test	f^{*5}	f^{*6}	f^{*7}	f^{*8}
Fréquence (Monobit)	0.401 (0.97)	0.924 (1.0)	0.779 (0.98)	0.883 (0.99)
Fréquence ds un bloc	0.574 (0.98)	0.062 (1.0)	0.978 (0.98)	0.964 (0.98)
Somme Cumulé*	0.598 (0.975)	0.812 (1.0)	0.576 (0.99)	0.637 (0.99)
Exécution	0.998 (0.99)	0.213 (0.98)	0.816 (0.98)	0.494 (1.0)
Longue exécution dans un bloc	0.085 (0.99)	0.971 (0.99)	0.474 (1.0)	0.574 (0.99)
Rang	0.994 (0.96)	0.779 (1.0)	0.191 (0.99)	0.883 (0.99)
Fourier rapide	0.798 (1.0)	0.595 (0.99)	0.739 (0.99)	0.595 (1.0)
Patron sans superposition*	0.521 (0.987)	0.494 (0.989)	0.530 (0.990)	0.520 (0.989)
Patron avec superposition	0.066 (0.99)	0.040 (0.99)	0.304 (1.0)	0.249 (0.98)
Statistiques universelles	0.851 (0.99)	0.911 (0.99)	0.924 (0.96)	0.066 (1.0)
Entropie approchée (m=10)	0.637 (0.99)	0.102 (0.99)	0.115 (0.99)	0.350 (0.98)
Suite aléatoire *	0.573 (0.981)	0.144 (0.989)	0.422 (1.0)	0.314 (0.984)
Suite aléatoire variante *	0.359 (0.968)	0.401 (0.982)	0.378 (0.989)	0.329 (0.985)
Série* (m=10)	0.469 (0.98)	0.475 (0.995)	0.473 (0.985)	0.651 (0.995)
Complexité linaire	0.129 (1.0)	0.494 (1.0)	0.062 (1.0)	0.739 (1.0)

TABLE 6.4 – Test de NIST pour les fonctions du tableau 6.2 selon les itérations généralisées

Test	g^{*5}	g^{*6}	f^{*7}	f^{*8}
Fréquence (Monobit)	0.236 (1.0)	0.867 (0.99)	0.437 (0.99)	0.911 (1.0)
Fréquence ds un bloc	0.129 (0.98)	0.350 (0.99)	0.366 (0.96)	0.657 (1.0)
Somme Cumulé*	0.903 (0.995)	0.931 (0.985)	0.863 (0.995)	0.851 (0.995)
Exécution	0.699 (0.98)	0.595 (0.99)	0.181 (1.0)	0.437 (0.99)
Longue exécution dans un bloc	0.009 (0.99)	0.474 (0.97)	0.816 (1.0)	0.051 (1.0)
Rang	0.946 (0.96)	0.637 (0.98)	0.494 (1.0)	0.946 (1.0)
Fourier rapide	0.383 (0.99)	0.437 (1.0)	0.616 (0.98)	0.924 (0.99)
Patron sans superposition*	0.466 (0.990)	0.540 (0.989)	0.505 (0.990)	0.529 (0.991)
Patron avec superposition	0.202 (0.96)	0.129 (0.98)	0.851 (0.99)	0.319 (0.98)
Statistiques universelles	0.319 (0.97)	0.534 (0.99)	0.759 (1.0)	0.657 (0.99)
Entropie approchée (m=10)	0.075 (0.97)	0.181 (0.99)	0.213 (0.98)	0.366 (0.98)
Suite aléatoire *	0.357 (0.986)	0.569 (0.991)	0.539 (0.987)	0.435 (0.992)
Suite aléatoire variante *	0.398 (0.989)	0.507 (0.986)	0.668 (0.991)	0.514 (0.994)
Série* (m=10)	0.859 (0.995)	0.768 (0.99)	0.427 (0.995)	0.637 (0.98)
Complexité linaire	0.897 (0.99)	0.366 (0.98)	0.153 (1.0)	0.437 (1.0)

TABLE 6.5 – Test de NIST pour les fonctions du tableau 6.2 selon les itérations unaires

Test	5 bits	6 bits	7 bits	8bits
Fréquence (Monobit)	0.289 (1.0)	0.437 (1.0)	0.678 (1.0)	0.153 (0.99)
Fréquence ds un bloc	0.419 (1.0)	0.971 (0.98)	0.419 (0.99)	0.275 (1.0)
Somme Cumulé*	0.607 (0.99)	0.224 (0.995)	0.645 (0.995)	0.901 (0.99)
Exécution	0.129 (0.99)	0.005 (0.99)	0.935 (0.98)	0.699 (0.98)
Longue exécution dans un bloc	0.514 (1.0)	0.739 (0.99)	0.994 (1.0)	0.834 (0.99)
Rang	0.455 (0.97)	0.851 (0.99)	0.554 (1.0)	0.964 (0.99)
Fourier rapide	0.096 (0.98)	0.955 (0.99)	0.851 (0.97)	0.037 (1.0)
Patron sans superposition*	0.534 (0.990)	0.524 (0.990)	0.508 (0.987)	0.515 (0.99)
Patron avec superposition	0.699 (0.99)	0.616 (0.95)	0.071 (1.0)	0.058 (1.0)
Statistiques universelles	0.062 (0.99)	0.071 (1.0)	0.637 (1.0)	0.494 (0.98)
Entropie approchée (m=10)	0.897 (0.99)	0.383 (0.99)	0.366 (1.0)	0.911 (0.99)
Suite aléatoire *	0.365 (0.983)	0.442 (0.994)	0.579 (0.992)	0.296 (0.993)
Suite aléatoire variante *	0.471 (0.978)	0.559 (0.992)	0.519 (0.987)	0.340 (0.995)
Série* (m=10)	0.447 (0.985)	0.298 (0.995)	0.648 (1.0)	0.352 (0.995)
Complexité linaire	0.005 (0.98)	0.534 (0.99)	0.085 (0.97)	0.996 (1.0)

TABLE 6.6 – Test de NIST pour l'algorithme de Mersenne Twister

IV

Application au masquage d'information

7

DES EMBARQUEMENTS PRÉSERVANT LE CHAOS

La propriété de transitivité des fonctions chaotiques est à l'origine du marquage de documents numériques : grâce à cette propriété, la marque est diffusée sur tout le support. Ainsi, de tout média, même tronqué, on peut la réextraire. Dans ce chapitre, le processus d'embarquement d'un message dans un média est formalisé en section 7.1. La sécurité des approches de watermarking est étudiée selon deux critères : probabiliste d'une part (section 7.2) et chaotique (section 7.3) d'autre part. Une proposition d'embarquement dans le domaine fréquentiel est abordée en section 7.4.

On remarque cependant que l'algorithme formalisé dans ces sections ne permet d'embarquer *in fine* qu'un bit qui est vrai si l'image est marquée et faux dans le cas contraire. Il ne permet pas d'extraire le contenu du message initial à partir de l'image marquée. La section 7.5 propose une solution à ce problème.

Les trois premières sections de ce chapitre sont une reformulation du chapitre 22 de [Guy10]. Elles ont été publiées à [BCG12b]. L'extension a quant à elle été publiée dans [BCF+13].

7.1/ PROCESSUS DE MARQUAGE BINAIRE

Par la suite, le message numérique qu'on cherche à embarquer est noté y et le support dans lequel se fait l'insertion est noté x.

Le processus de marquage est fondé sur les itérations unaires d'une fonction selon une stratégie donnée. Cette fonction et cette stratégie sont paramétrées par un entier naturel permettant à la méthode d'être applicable à un média de n'importe quelle taille. On parle alors respectivement de *mode* et d'*adapteur de stratégies*

7.1.1/ EMBARQUEMENT

Définition 9 (Mode)**.** *Soit* N *un entier naturel. Un mode est une application de* $\mathbb{B}^{\mathrm{N}}$ *dans lui même.*

Définition 10 (Adapteur de Stratégie)**.** *Un* adapteur de stratégie *est une fonction* $\mathcal{S}$ *de* $\mathbb{N}$ *dans l'ensemble des séquences d'entiers qui associe à chaque entier naturel* N *la suite*

$S \in [\mathsf{N}]^{\mathbb{N}}$.

On définit par exemple l'adapteur CIIS (*Chaotic Iterations with Independent Strategy*) paramétré par $(K, y, \alpha, l) \in [0,1] \times [0,1] \times]0, 0.5[\times \mathbb{N}$ qui associe à chaque entier $\mathsf{N} \in \mathbb{N}$ la suite $(S^t)^{t \in \mathbb{N}}$ définie par :

- $K^0 = bin(y) \oplus bin(K)$: K^0 est le nombre binaire (sur 32 bits) égal au ou exclusif (xor) entre les décompositions binaires sur 32 bits des réels y et K (il est aussi compris entre 0 et 1),
- $\forall t \leqslant l, K^{t+1} = F(K^t, \alpha)$,
- $\forall t \leqslant l, S^t = \lfloor \mathsf{N} \times K^t \rfloor + 1$,
- $\forall t > l, S^t = 0$,

où F est la fonction chaotique linéaire par morceau [SQW+01]. Les paramètres K et α de cet adapteur de stratégie peuvent être vus comme des clefs. On remarque que cette stratégie est unaire.

On peut attribuer à chaque bit du média hôte x sa valeur d'importance sous la forme d'un réel. Ceci se fait à l'aide d'une fonction de signification.

Définition 11 (Fonction de signification)**.** *Une* fonction de signification *est une fonction u qui à toute séquence finie de bit x associe la séquence $(u^k(x))$ de taille $\mid x \mid$ à valeur dans les réels. Cette fonction peut dépendre du message y à embarquer, ou non.*

Pour alléger le discours, par la suite, on notera $(u^k(x))$ pour (u^k) lorsque cela n'est pas ambigüe. Il reste à partitionner les bits de x selon qu'ils sont peu, moyennement ou très significatifs.

Définition 12 (Signification des bits)**.** *Soit u une fonction de signification, m et M deux réels t.q. $m < M$. Alors : u_M, u_m et u_p sont les vecteurs finis respectivement des* bits les plus significatifs (MSBs) *de x,* bits les moins significatifs (LSBs) *de x* bits passifs *de x définis par :*

$$\begin{aligned} u_M &= \left(k \;\middle|\; k \in \mathbb{N} \text{ et } u^k \geqslant M \text{ et } k \leq \mid x \mid\right) \\ u_m &= \left(k \;\middle|\; k \in \mathbb{N} \text{ et } u^k \leq m \text{ et } k \leq \mid x \mid\right) \\ u_p &= \left(k \;\middle|\; k \in \mathbb{N} \text{ et } u^k \in]m; M[\text{ et } k \leq \mid x \mid\right) \end{aligned}$$

On peut alors définir une fonction de décomposition puis de recomposition pour un hôte x :

Définition 13 (Fonction de décomposition)**.** *Soit u une fonction de signification, m et M deux réels t.q $m < M$. Tout hôte x peut se décomposer en $(u_M, u_m, u_p, \phi_M, \phi_m, \phi_p)$ avec*

- *u_M, u_m, et u_p construits comme à la définition ,*
- $\phi_M = \left(x^{u_M^1}, x^{u_M^2}, \ldots, x^{u_M^{|u_M|}}\right)$,
- $\phi_m = \left(x^{u_m^1}, x^{u_m^2}, \ldots, x^{u_m^{|u_m|}}\right)$,
- $\phi_p = \left(x^{u_p^1}, x^{u_p^2}, \ldots, x^{u_p^{|u_p|}}\right)$.

La fonction qui associe $(u_M, u_m, u_p, \phi_M, \phi_m, \phi_p)$ pour chaque hôte x est la fonction de décomposition*, plus tard notée dec(u, m, M) puisqu'elle est paramétrée par u, m et M.*

Définition 14 (Recomposition)**.** *Soit un sextuplet $(u_M, u_m, u_p, \phi_M, \phi_m, \phi_p) \in \mathfrak{N} \times \mathfrak{N} \times \mathfrak{N} \times \mathfrak{B} \times \mathfrak{B} \times \mathfrak{B}$ tel que*

— *les ensembles* u_M, u_m *et* u_p *forment une partition de* [N] ;
— $|u_M| = |\varphi_M|$, $|u_m| = |\varphi_m|$ *et* $|u_p| = |\varphi_p|$.

Soit la base canonique sur l'espace vectoriel $\mathbb{R}^{|x|}$ *composée des vecteurs* $e_1, \ldots, e_{|x|}$. *On peut construire le vecteur*

$$x = \sum_{i=1}^{|u_M|} \varphi_M^i . e_{u_M^i} + \sum_{i=1}^{|u_m|} \varphi_m^i . e_{u_m^i} + \sum_{i=1}^{|u_p|} \varphi_p^i . e_{u_p^i}$$

La fonction qui associe x *à chaque sextuplet* $(u_M, u_m, u_p, \phi_M, \phi_m, \phi_p)$ *défini comme ci-dessus est appelée* fonction de recomposition.

Un embarquement consiste à modifier les valeurs de ϕ_m (de x) en tenant compte de y. Cela se formalise comme suit :

Définition 15 (Embarquement de message)**.** *Soit une fonction de décomposition dec*(u, m, M), x *un support,* $(u_M, u_m, u_p, \phi_M, \phi_m, \phi_p)$ *son image par dec*(u, m, M), *et* y *un média numérique de taille* $|u_m|$. *Le média* z *résultant de l'embarquement d'*y *dans* x *est l'image de* $(u_M, u_m, u_p, \phi_M, y, \phi_p)$ *par la fonction de recomposition rec.*

On peut étendre l'algorithme dhCI [GFB10] d'embarquement de message comme suit :

Définition 16 (Embarquement dhCI étendu)**.** *Soit dec*(u, m, M) *une fonction de décomposition,* f *un mode,* $\mathcal{S}$ *un adapteur de stratégie,* x *un hôte,* $(u_M, u_m, u_p, \phi_M, \phi_m, \phi_p)$ *son image par dec*(u, m, M), q *un entier naturel positif et* y *un média numérique de taille* $l = |u_m|$.

L'algorithme d'embarquement de message associe à chaque couple (x, y) *le média* z *résultat de l'embarquement de* $\hat{y}$ *dans* x, *t. q. :*

— *le mode* f *est instancié avec le paramètre* $l = |u_m|$, *engendrant la fonction* $f_l : \mathbb{B}^l \to \mathbb{B}^l$;
— *l'adapteur de stratégie* $\mathcal{S}$ *est instancié avec le paramètre* y, *engendrant une stratégie* $S_y \in [l]$;
— *on itère la fonction* G_{f_l} *en prenant la configuration initiale* (S_y, ϕ_m) *selon le schéma défini à l'équation (3.3).*
— $\hat{y}$ *est le second membre du* $q^{ème}$ *terme obtenu.*

La figure 7.1 synthétise la démarche.

7.1.2/ DÉTECTION D'UN MÉDIA MARQUÉ

On caractérise d'abord ce qu'est un média marqué selon la méthode énoncée à la section précédente. On considère que l'on connaît la marque à embarquer y, le support x et que l'on a face à soi un média z.

Définition 17 (Média marqué)**.** *Soit dec*(u, m, M) *une fonction de décomposition,* f *un mode,* $\mathcal{S}$ *un adapteur de stratégie,* q *un entier naturel strictement positif,* y *un média digital et soit* $(u_M, u_m, u_p, \phi_M, \phi_m, \phi_p)$ *l'image par dec*(u, m, M) *du média* x. *Alors,* z *est* marqué *avec* y *si l'image par dec*(u, m, M) *de* z *est* $(u_M, u_m, u_p, \phi_M, \hat{y}, \phi_p)$, *où* $\hat{y}$ *est le second membre de* $G_{f_l}^q(S_y, \phi_m)$.

7.2/ ANALYSE DE SÉCURITÉ (PROBABILISTES)

Récemment [CFF05, PFPGC06] ont proposé des classes de sécurité pour le marquage d'information. Parmi celles-ci, la stego-sécurité a été au centre des travaux puisqu'elle représente la classe la plus élevée dans le contexte où l'attaquant n'a accès qu'à l'hôte marqué z.

Cette définition probabiliste est rappelée ci-après. Soit $\mathbb{K}$ un ensemble de clefs, $p(X)$ un modèle probabiliste de N_0 hôtes, et $p(Y|K)$ le modèle probabiliste de N_0 contenus marqués avec la même clé K et le même algorithme d'embarquement.

Définition 18 (Stégo-Sécurité [CB08])**.** *La fonction d'embarquement est* stégo-sécure *si la propriété* $\forall K \in \mathbb{K}, p(Y|K) = p(X)$ *est établie.*

Il a déjà été démontré [Guy10, GFB10] que l'algorithme de marquage dont le mode est la fonction négation est stégo-sécure. Un des intérêts de l'algorithme présenté ici est qu'il est paramétré par un mode. Lorsque celui-ci a les même propriétés que celles vues pour la création de PRNG (*i.e.* graphe des itérations fortement connexes et matrice de Markov doublement stochastique), on a un marquage qui peut être rendu stégo-sécure à ε près, ce que précise le théorème suivant. La preuve de ce théorème est donnée en annexes D.

Théorème 25 (ε-stego sécurité)**.** *Soit* ε *un nombre positif,* l *un nombre de LSBs,* $X \sim \mathbf{U}\left(\mathbb{B}^l\right)$*, un adapteur de stratégie uniformément distribué indépendant de* X f_l *un mode tel que* $\mathrm{GIU}(f_l)$ *est fortement connexe et la matrice de Markov associée à* f_l *est doublement stochastique. Il existe un nombre* q *d'itérations tel que* $|p(Y|K) - p(X)| < \varepsilon$*.*

7.3/ ANALYSE DE SÉCURITÉ (CHAOS)

On rappelle uniquement la définition de chaos-sécurité introduite dans [Guy10].

Définition 19 (Chaos-sécurité)**.** *Un schéma de marquage* S *est chaos-sécure sur un espace topologique* $(\mathcal{X}, \tau)$ *si sa version itérative a un comportement chaotique sur celui-ci.*

Tout repose ainsi sur la capacité que l'on a à produire des fonctions dont le graphe des itérations unaires sera fortement connexe. Ceci a déjà été traité au chapitre 3. La seule complexité est l'adaptabilité de la fonction au nombre l de LSBs.

On considère par exemple le mode $f_l : \mathbb{B}^l \to \mathbb{B}^l$ t.q. le $i^{\text{ème}}$ composant est défini par

$$f_l(x)_i = \begin{cases} \overline{x_i} \text{ si } i \text{ est impair} \\ x_i \oplus x_{i-1} \text{ si } i \text{ est pair} \end{cases} \tag{7.1}$$

on peut déduire immédiatement du théorème 15 (chap. 3) que le graphe $\mathrm{GIU}(f_l)$ est fortement connexe. La preuve de double-stochasticité de la matrice associée à f_l est donnée en annexe D.2. On dispose ainsi d'un nouvel algorithme de marquage ε-stégo-sécure et chaos-sécure.

7.4/ APPLICATIONS AUX DOMAINES FRÉQUENTIELS

Le schéma d'algorithme présenté dans ce chapitre a été appliqué au marquage d'images dans les coefficients DCT et les DWT.

7.4.1/ FONCTION DE SIGNIFICATION POUR L'EMBARQUEMENT DANS LES DCT

On considère un hôte x de taille $H \times L$ dans le domaine fréquentiel DCT. Dans chaque bloc de taille 8×8, à chaque bit la fonction de signification u associe

- 1 si c'est un bit apparaissant dans la représentation binaire de la valeur d'un coefficient dont les coordonnées appartiennent à $\{(1,1),(2,1),(1,2)\}$,
- 1 si c'est un bit apparaissant dans la représentation binaire de la valeur d'un coefficient dont les coordonnées appartiennent à $\{(3,1),(2,2),(1,3)\}$ et qui n'est pas un des trois bits de poids faible de cette représentation,
- -1 si c'est un bit apparaissant dans la représentation binaire de la valeur d'un coefficient dont les coordonnées appartiennent à $\{(3,1),(2,2),(1,3)\}$ et qui est un des trois bits de poids faible de cette valeur,
- 0 sinon.

Le choix de l'importance de chaque coefficient est défini grâce aux seuils $(m, M) = (-0.5, 0.5)$ permettant d'engendrer les MSBs, LSBs, et bits passifs.

7.4.2/ FONCTION DE SIGNIFICATION POUR L'EMBARQUEMENT DANS LES DWT

On considère un hôte dans le domaine des DWT. La fonction de signification se concentre sur les seconds niveaux de détail (*i.e.*, LH2, HL2 et HH2). Pour chaque bit, on dit qu'il est peu significatif si c'est un des trois bits de poids faible d'un coefficient de LH2, HL2 ou de HH2. Formellement à chaque bit la fonction de signification u associe

- 1 si c'est un bit apparaissant dans la représentation binaire de la valeur d'un coefficient de type LL2,
- 1 si c'est un bit apparaissant dans la représentation binaire de la valeur d'un coefficient de type LH2, HL2, HH2 et qui n'est pas un des trois bits de poids faible de cette représentation,
- 0 si c'est un bit apparaissant dans la représentation binaire de la valeur d'un coefficient de type LH2, HL2, HH2 et qui est un des trois bits de poids faible de cette représentation,
- -1 sinon.

Le choix de l'importance de chaque coefficient est encore défini grâce aux seuils $(m, M) = (-0.5, 0.5)$ permettant d'engendrer les MSBs, LSBs, et bits passifs.

7.4.3/ ETUDE DE ROBUSTESSE

Cette partie synthétise une étude de robustesse de la démarche présentée ci-avant. Dans ce qui suit, dwt(neg), dwt(fl), dct(neg), dct(fl) correspondant respectivement aux embarquements en fréquentiels dans les domaines DWT et DCT avec le mode de négation et celui issu de la fonction f_l détaillée à l'équation 7.3.

A chaque série d'expériences, un ensemble de 50 images est choisi aléatoirement de la base du concours BOSS [PFB10a]. Chaque hôte est une image en 512×512 en niveau de gris et la marque y est une suite de 4096 bits. La résistance à la robustesse est évaluée en appliquant successivement sur l'image marquée des attaques de découpage, de compression, de transformations géométriques. Si les différences entre $\hat{y}$ and $\varphi_m(z)$. sont en dessous d'un seuil (que l'on définit), l'image est dite marquée (et non marquée dans le cas contraire). Cette différence exprimée en pourcentage est rappelée pour chacune des attaques à la figure 7.2.

7.4.4/ ÉVALUATION DE L'EMBARQUEMENT

Pour évaluer le seuil qui permet de dire avec la plus grande précision si une image est marquée ou non, nous avons appliqué la démarche suivante. A partir d'un ensemble de 100 images du challenge BOSS, les trois ensembles suivants sont construits : celui des images marquées W, celui contenant des images marquées puis attaquée *WA*, et celui des images uniquement attaquées A. Les attaques sont choisies parmi celles données ci dessus.

Pour chaque entier t entre 5 et 55 et chaque image $x \in WA \cup A$, on calcule la différence entre $\hat{y}$ et $\varphi_m(z)$. L'image est dite marquée si cette différence est en dessous du seuil t considéré

- si elle est dite marquée et si x appartient à *WA* c'est un vrai cas positif (TP) ;
- si elle est dite non marquée et si x appartient cependant à *WA* c'est un faux cas négatif (FN) ;
- si elle est dite marquée et si x appartient cependant à *A* c'est un faux cas positif (FP) ;
- enfin si elle est dite non marquée et si x appartient à *A* c'est un vrai cas négatif (TN).

La courbe ROC construite à partir des points de coordonnées (TP,FP) issus de ces seuils est donnée à la figure 7.3. Pour la fonction f_l et pour la fonction négation respectivement, la détection est optimale pour le seuil de 45% correspondant au point (0.01, 0.88) et pour le seuil de 46% correspondant au point (0.04, 0.85) dans le domaine DWT. Pour les deux modes dans le domaine DCT, la détection est optimale pour le seuil de 44% (correspondant aux points (0.05, 0.18) et (0.05, 0.28)). On peut alors donner des intervalles de confiance pour les attaques évaluées. L'approche est résistante :

- à tous les découpages où le pourcentage est inférieur à 85% ;
- aux compression dont le ratio est supérieur à 82% dans le domaine DWT et 67% dans celui des DCT ;
- aux modifications du contraste lorsque le renforcement est dans $[0.76, 1.2]$ dans le domaine DWT et $[0.96, 1.05]$ dans le domaine DCT ;
- à toutes les rotations dont l'angle est inférieur à 20 degrés dans le domaine DCT et celles dont l'angle est inférieur à 13 degrés dans le domaine DWT.

7.5/ EMBARQUONS DAVANTAGE QU'1 BIT

L'algorithme présenté dans les sections précédentes ne permet de savoir, *in fine*, que si l'image est marquée ou pas. Cet algorithme ne permet pas de retrouver le contenu de la

marque à partir de l'image marquée. C'est l'objectif de l'algorithme présenté dans cette section et introduit dans [FGB11]. Pour des raisons de lisibilité, il n'est pas présenté dans le formalisme de la première section et est grandement synthétisé. Il a cependant été prouvé comme étant chaos-sécure [FGB11].

Commençons par quelques conventions de notations :

- $\mathbb{S}_{\mathsf{K}}$ est l'ensemble des stratégies unaires sur $[k]$;
- $m^0 \in \mathbb{B}^{\mathsf{P}}$ est un vecteur de P bits représentant la marque ;
- comme précédemment, $x^0 \in \mathbb{B}^{\mathsf{N}}$ est le vecteurs des N bits sélectionnés où la marque est embarquée.
- $S_p \in \mathbb{S}_{\mathsf{N}}$ est la *stratégie de place* et définit quel élément de x est modifié à chaque itération ;
- $S_c \in \mathbb{S}_{\mathsf{P}}$ est la *stratégie de choix* qui définit quel indice du vecteur de marque est embarqué à chaque itération ;
- $S_m \in \mathbb{S}_{\mathsf{P}}$ est la *stratégie de mélange* qui précise quel élément de la marque est inversé à chaque itération.

Le processus itératif modifiant x est défini comme suit. Pour chaque $(n, i, j) \in \mathbb{N}^* \times [\![0; \mathsf{N}-1]\!] \times [\![0; \mathsf{P}-1]\!]$, on a :

$$\begin{cases} x_i^n = \begin{cases} x_i^{n-1} & \text{si } S_p^n \neq i \\ m_{S_c^n}^{n-1} & \text{si } S_p^n = i. \end{cases} \\ \\ m_j^n = \begin{cases} m_j^{n-1} & \text{si } S_m^n \neq j \\ \\ \overline{m_j^{n-1}} & \text{si } S_m^n = j. \end{cases} \end{cases}$$

où $\overline{m_j^{n-1}}$ est la négation booléenne de m_j^{n-1}. On impose de plus la contrainte suivante. Soit $\mathfrak{I}(S_p) = \{S_p^1, S_p^2, \ldots, S_p^l\}$ l'ensemble de cardinalité $k \leq l$ (les doublons sont supprimés) qui contient la liste des indices i, $1 \leq i \leq \mathsf{N}$, tels que x_i a été modifié. On considère $\mathfrak{I}(S_c)_{|D} = \{S_c^{d_1}, S_c^{d_2}, \ldots, S_c^{d_k}\}$ où d_i est la dernière date où l'élément $i \in \mathfrak{I}(S_p)$ a été modifié. Cet ensemble doit être égal à $[\![0; \mathsf{P}-1]\!]$.

Pour peu que l'on sache satisfaire la contrainte précédente, on remplace x par $x^l \in \mathbb{B}^{\mathsf{N}}$ dans l'hôte et on obtient un contenu marqué.

Sans attaque, le schéma doit garantir qu'un utilisateur qui dispose des bonnes clefs de création des stratégies est capable d'extraire une marque et que celle-ci est la marque insérée. Ceci correspond respectivement aux propriétés de complétude et de correction de l'approche. L'étude de ces propriétés est l'objectif de la section qui suit.

7.5.1/ CORRECTION ET COMPLÉTUDE DU SCHÉMA

On ne donne ici que le théorème. La preuve est placée en annexes D.3.

Théorème [26] (Correction et complétude du marquage)**.** *La condition de l'algorithme de marquage est nécessaire et suffisante pour permettre l'extraction du message du média marqué.*

Sous ces hypothèse, on peut donc extraire un message. De plus, le cardinal k de $\mathfrak{I}(S_p)$ est supérieur ou égal à P. Ainsi le bit j du message original m^0 peut être embarqué plusieurs fois dans x^l. Or, en comptant le nombre de fois où ce bit a été inversé dans

S_m, la valeur de m_j peut se déduire en plusieurs places. Sans attaque, toutes ces valeurs sont identiques et le message est obtenu immédiatement. Si attaque il y a, la valeur de m_j peut s'obtenir en prenant la valeur moyenne de toutes les valeurs obtenues.

7.5.2/ DÉTECTER SI LE MÉDIA EST MARQUÉ

On considère un média y marqué par un message m. Soit y' une version altérée de y, c.-à-d. une version où certains bits ont été modifiés et soit m' le message extrait de y'.

Pour mesurer la distance entre m' et m, on considère respectivement M et M' l'ensemble des indices de m et de m' où m_i vaut 1 et ou m'_1 vaut 1.

Beaucoup de mesures de similarité [YK50, And73, RBBM00], dépendent des ensembles a, b, c et d définis par $a = |M \cap M'|$, $b = |M \setminus M'|$, $c = |M' \setminus M|$ et $d = |\overline{M} \cap \overline{M'}|$

Selon [RDBM03] la mesure de Fermi-Dirac S_{FD} est celle dont le pouvoir de discrimination est le plus fort, c.-à-d. celle qui permet la séparation la plus forte entre des vecteurs corrélés et des des vecteurs qui ne le sont pas. La distance entre m et m' est construite selon cette mesure et produit un réel dans $[0; 1]$. Si elle est inférieure à un certain seuil (à définir), le média y' est déclaré comme marqué et le message doit pouvoir être extrait.

7.5.3/ ETUDE DE ROBUSTESSE

La méthode d'expérimentation de robustesse appliquée à la section précédente pourrait être réappliquée ici et nous pourrions obtenir, grâce aux courbes de ROC une valeur seuil pour déterminer si une marque est présente ou pas. Dans [BCF+13], nous n'avons cependant pas poussé la démarche plus loin que dans la direction de l'embarquement dans les bits de poids faible en spatial et l'on sait que ceci est particulièrement peu robuste.

7.6/ CONCLUSION

Grâce à la formalisation du processus de watermarking par itérations discrètes, nous avons pu dans ce chapitre montrer que le processus possédait les propriétés attendues, à savoir stego-sécurité, chaos sécurité et une robustesse relative. Pour étendre le champ applicatif, nous avons proposé un second algorithme permettant de particulariser la marque à embarquer et donc à extraire. Le chapitre suivant s'intéresse au marquage, mais dans un autre domaine que celui des images.

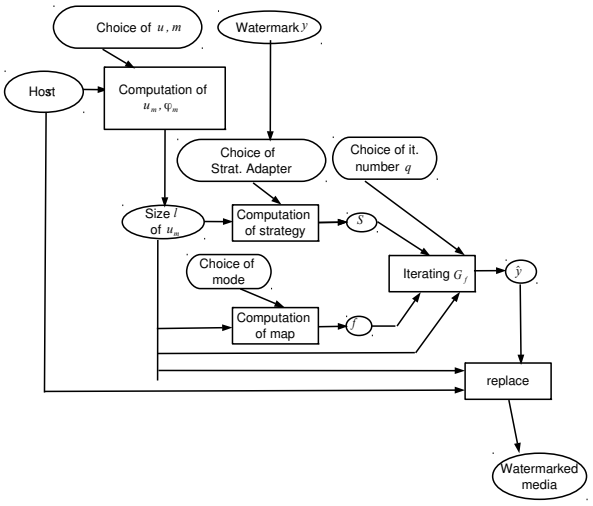
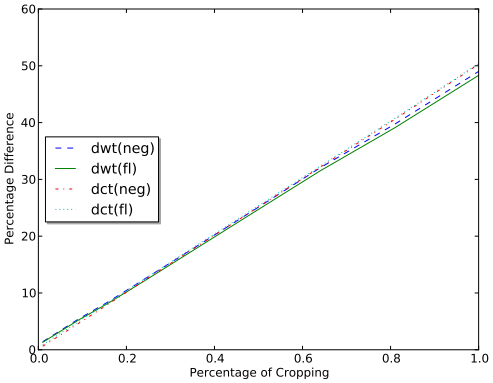
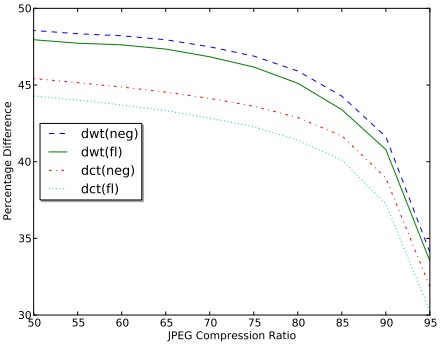


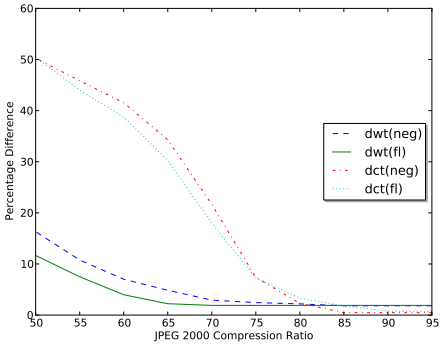
FIGURE 7.1 – Le schéma de marquage dhCI



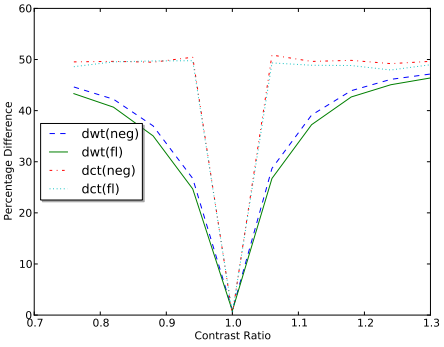
(a) Découpage



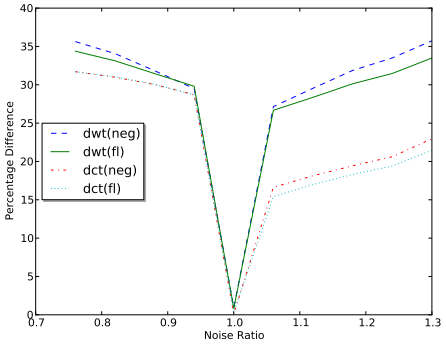
(b) Compression JPEG



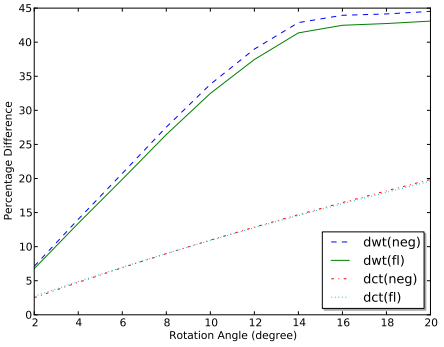
(c) Compression JPEG 2000



(d) Modification du contraste



(e) Accentuation des bords



(f) Rotation

FIGURE 7.2 – Illustration de la robustesse

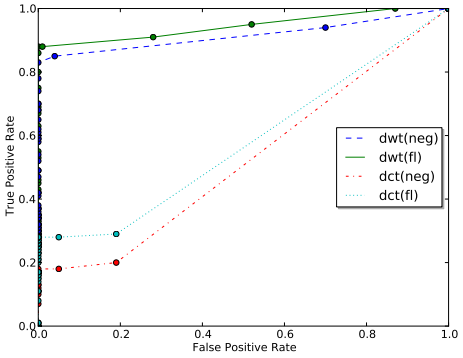


FIGURE 7.3 – Courbes ROC de seuils de détection

8

UNE DÉMARCHE DE MARQUAGE DE PDF

En étudiant les schémas de watermarking, nous avons constaté que très peu de travaux ciblaient les documents PDF qui représentent cependant une part non anecdotique des données échangées en ligne. Parmi ces travaux, [PD08] propose la modification du nombre d'espaces entre les mots ou entre les paragraphes. Similairement, les auteurs de [LT10] ajoutent des caractères invisibles dans le document. En supprimant ces espaces ou caractères invisibles, la marque s'enlève facilement. Dans [PD08], les auteurs modifient de manière imperceptible le positionnement des caractères. D'autres éléments de positionnement sont intégrés dans [WT08]. Une attaque qui modifierait aléatoirement de manière faible ces positions détruirait la marque dans les deux cas. La quantification (au sens du traitement du signal) est une réponse à ces attaques : des positions modifiées de manière mal intentionnée peuvent grâce à cette démarche être rapprochées (abstraites) en des positions préétablies et conserver ainsi leur information et donc la marque. STDM [CW01] est une instance de ces schémas de marquage.

Ce chapitre présente une application de STDM au marquage de documents PDFs. La première section fournit quelques rappels sur la STDM. Le schéma basé sur cette approche est présenté à la section 8.2. Finalement, la démarche expérimentale permettant de trouver un compromis entre robustesse et qualité visuelle est présentée à la section 8.3. Ce travail a été publié dans [BDCC15].

8.1/ RAPPELS SUR LA SPREAD TRANSFORM DITHER MODULATION

Les paramètres de ce schéma sont

- le facteur de quantification Δ qui est un réel positif ; plus Δ est grand, plus la distorsion peut être importante ;
- le niveau d'indécision d_0 qui est un réel dans $[-\frac{\Delta}{2}, \frac{\Delta}{2}]$; plus ce nombre a une valeur absolue élevée, plus les erreurs peuvent être corrigées ; on définit d_1 par

$$d_1 = \begin{cases} d_0 + \Delta/2, & \text{si} \quad d_0 < 0 \\ d_0 - \Delta/2, & \text{sinon} \end{cases}$$

- un nombre L d'éléments dans lequel chaque bit de la marque est embarqué ;

— un vecteur p de projection de taille L.

Soit donc x un vecteur de taille L dans lequel on souhaite embarquer le bit $m \in \{0, 1\}$. Ce vecteur est remplacé par x' défini par

$$x' = f(x, m) = x + ((\lfloor(\frac{(x^T p) - d_m}{\Delta})\rfloor\Delta + d_m) \ - x^T p)p \tag{8.1}$$

Avec les mêmes paramètres Δ, d_0 , L et p le message $\hat{m}$ extrait de x' de taille L est défini par :

$$\hat{m} = arg \min_{m\in\{0,1\}} \mid x'^T p - f(x, m) \mid \tag{8.2}$$

Les auteurs de [CW01] ont montré que la variance de l'erreur est égale à $D_s = \Delta^2/12L$ lorsque chacun des L éléments de x suit une distribution uniforme $U(\Delta)$. Tous les éléments sont en place pour embarquer une marque dans un fichier PDF selon le schéma STDM.

8.2/ APPLICATION AU MARQUAGE DE DOCUMENTS PDF

On détaille successivement comment insérer une marque dans un document PDF, puis comment l'extraire.

8.2.1/ INSERTION DE LA MARQUE

On cherche à ajouter à un document PDF une marque m de k bits déjà codée (cryptée, correction d'erreurs incluse). L'insertion de celle-ci dans le document s'effectue en quatre étapes.

On considère comme fixés les paramètres Δ, d_0 , L et la manière de construire le vecteur p pour ce L donné.

1. Le vecteur hôte x de taille N est constitué de l'abscisse (flottante) de chaque caractère rencontré dans le document PDF. La dimension L est calculée comme la partie entière de N/k.
2. Un générateur pseudo-aléatoire (initialisé par une clef) construit k ensembles M_1, ..., M_k de taille L mutuellement disjoints dans $[1, N]$. Ainsi $\bigcup_{1\leq i\leq k} M_i \subseteq [N]$.
3. Pour chacun des ensembles M_i, $1 \leq i \leq k$, de l'étape précédente, le vecteur $\dot{x} = (x_{j_1}, \ldots, x_{j_L})$, est construit où $\{j_1, \ldots, j_L\} = M_i$. Le vecteur $\dot{x'} = f(\dot{x}, m_i)$ est construit selon l'équation (8.1). Dans x, chacun des $x_{j_1}, \ldots, x_{j_L}$ est remplacé par $\dot{x'}_{j_1}, \ldots, \dot{x'}_{j_L}$.
4. L'abscisse de chaque caractère est ainsi redéfini selon le nouveau vecteur de positions x'.

Voyons comment extraire une marque d'un document PDF.

8.2.2/ EXTRACTION DE LA MARQUE

On considère comme connue la taille de la marque : c'est k bits. Les paramètres Δ, d_0 et la manière de construire p en fonction de L sont les mêmes qu'à l'étape précédente d'insertion de marque.

1. on récupère le vecteur x' (de taille N lui aussi) des abscisse des caractères du document PDF comme dans la phase d'insertion. la valeur de L est définie comme précédemment.

2. le même générateur pseudo-aléatoire (initialisé avec la même clef) construit les k mêmes ensembles $M_1, \ldots, M_k$ de taille L mutuellement disjoints dans $[1, N]$.

3. Pour chacun des ensembles M_i, $1 \leq i \leq k$, de l'étape précédente, le vecteur $\dot{x'} = (x'_{j_1}, \ldots, x'_{j_L})$, est construit où $\{j_1, \ldots, j_L\} = M_i$. Le bit $\hat{m}_i$ est défini selon l'équation (8.2) en remplaçant x' par $\dot{x'}$.

8.3/ EXPÉRIMENTATIONS

Le schéma de marquage est paramétré par Δ, d_0 et la manière de construire le vecteur p pour une taille L. Les travaux réalisés se sont focalisés sur l'influence du paramètre $D_S = \frac{\Delta^2}{12L}$ dans l'algorithme en satisfaisant les deux contraintes antagonistes de fournir une marque suffisamment robuste et suffisamment transparente. On cherche deux réels a et b tels que a et b correspondent respectivement au seuil maximum pour être transparent et au seuil minimum pour être robuste. Les études de perceptibilité doivent permettre de déterminer a tandis que celles sur la robustesse devront fixer le seuil b. Finalement, les contraintes précédentes seront satisfaites si et seulement si $a > b$ et $D_s \in [b, a]$.

Concernant la transparence, les expériences présentées dans l'article [BDCC15] ont consisté en choisir un texte d'un nombre fixe de caractères n dans lequel doit être embarqué une marque de taille fixe k. En faisant varier la valeur de Δ, nous avons remarqué que la valeur $a = 0,01335$ est le seuil au delà duquel il est visuellement possible de remarquer une différence entre le document original et le document marqué.

Il nous reste à détailler les expériences d'étude de robustesse de la démarche. Comme dans l'évaluation de la transparence, il s'est agi de faire varier le paramètre Δ. Pour chacune de ces valeurs, le document a été altéré selon un flou gaussien (de paramètre 0,1 et 0,25) et une attaque de type poivre et sel (de paramètre 0,1 et 0,25 aussi). Le rapport entre le nombre de bits erronés par rapport au nombre total de bits (nommé BER ci-après) après l'extraction du message est alors calculé. Le facteur de quantification a été choisi entre 0.1 et 10. L'expérience a été répétée 500 fois et les moyennes sont représentées à la figure 8.1. Sur cette figure, on constate que pour peu que la quantification Δ soit supérieure à 1, le taux d'erreur est inférieur à 12,5%. Ce taux peut être corrigé par un code correcteur usuel. Avec les paramètres de l'expérimentation, cela revient à considérer un seuil $b = 0,00214$. Ces expériences ont ainsi pu valider l'existence de seuils de distorsion permettant d'avoir une méthode à la fois robuste et transparente.

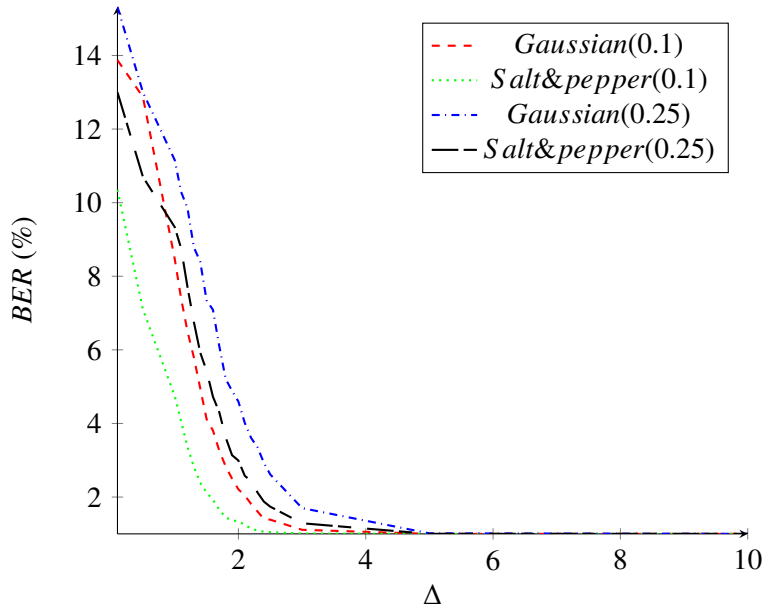


FIGURE 8.1 – Représentation du BER pour des attaques de type flou gaussien et poivre et sel

8.4/ CONCLUSION

Ce travail a présenté une démarche outillée basée sur la Spread Transform Dither Modulation permettant d'embarquer une marque dans un document PDF. Les éléments modifiés sont les abscisses des caractères présents dans le document.

Deux des propriétés essentielles des algorithmes de marquage ont été étudiées : la transparence et la robustesse. La notion d'intervalle de distorsion acceptable a été définie et calculée sur un exemple jouet.

9

UNE DÉMARCHE PLUS CLASSIQUE DE DISSIMULATION : STABYLO

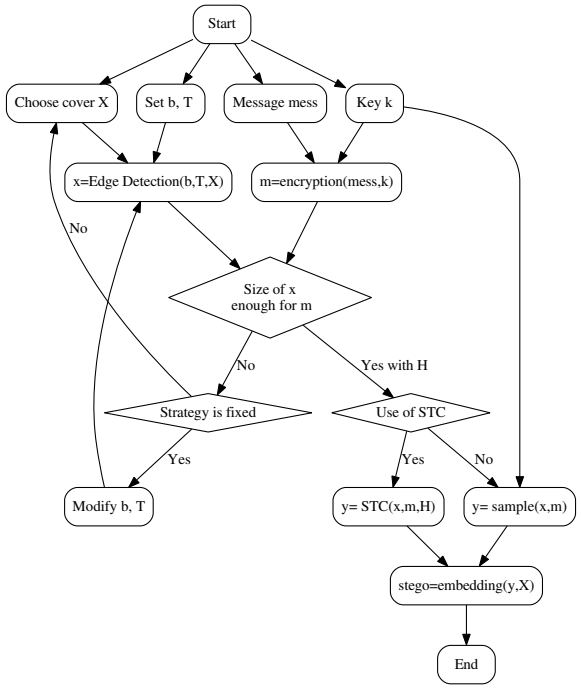
Dans cette partie, on s'intéresse toujours à insérer un message dans une image hôte. Si l'objectif des exemples précédents était de marquer l'hôte de manière robuste (et peu visible), c'est ici l'imperceptibilité qui est visée. La *stéganographie* est la famille des démarches qui visent à embarquer un message dans un hôte sans que l'on puisse discerner un hôte vierge d'une image contenant un message. Les outils les plus récents et les plus efficaces de cette famille sont HUGO [PFB10b], WOW [HF12] et UNIWARD [HFD14]. Pour détecter la présence ou non d'un message dans une image, on peut demander l'oracle à un *stéganalyseur* [LHS08, Ker05, FK12]. Usuellement, un outil de cette famille, après une démarche d'apprentissage, classifie les images en fonction de caractéristiques numériques.

A partir de caractéristiques de voisinage nommées SPAM [PBF10], HUGO mesure la distorsion qui serait induite par la modification de chaque pixel. Similairement, WOW et UNIWARD construisent une carte de distorsion mais celle-ci est issue de caractéristiques directionnelles calculées à partir d'ondelettes. A partir de ces cartes de distorsion, chacun de ces algorithmes sélectionne les pixels dont les modifications induisent la distorsion la plus faible possible. Ceci revient à définir une fonction de signification u. La complexité du schéma de stéganographie est peu ou prou celle du calcul de cette carte, et elle est élevée dans le cas de ces algorithmes. Nous avons proposé un algorithme [CCG15] de complexité beaucoup plus faible et dont la détectabilité est satisfaisante. Ce chapitre détaille les clefs de ce schéma

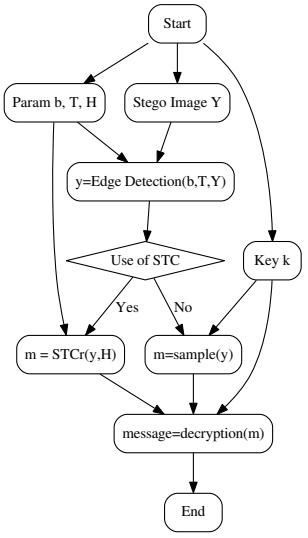
9.1/ PRÉSENTATION DE L'APPROCHE

Le diagramme de flux donné à la Fig. 9.1 résume l'approche du schéma STABYLO (pour STeganography with Adaptive, Bbs, binarY embedding at LOw cost). L'embarquement est synthétisé à la Fig. 9.1(a) et l'extraction à la Fig. 9.1(b).

La sécurité de l'encryptage est garantie par le système asymétrique de Blum-Goldwasser [BG85] basé sur le PRNG Blum Blum Shub [BBS83]. Ainsi, à partir d'une clef k et un message *mess*, ce cryptosystem construit le message m.



(a) Data Embedding



(b) Data Extraction

FIGURE 9.1 – Présentation générale de STABYLO

9.1.1/ UN EMBARQUEMENT DANS LES BORDS

L'idée d'embarquer dans les bords d'une image repose sur le fait que les pixels de ceux-ci représentent déjà une rupture de continuité entre pixels voisins. Une faible modification de ceux-ci n'aurait donc pas un grand impact sur la qualité de l'image, condition nécessaire lorsqu'on prétend être indétectable.

STABYLO est basé sur les filtres de Canny [Can86], comme démarche de détection de bords retenue pour sa complexité faible et ses possibilités d'implantation sur plusieurs supports (GPU, FPGA notamment). Rien n'interdirait cependant de l'appliquer à d'autres approches de détection de bord (Sobel, à base de logique floue [KF11],...). Cette détection de bords ne considère que les b bits les plus significatifs (pratiquement b vaut 6 ou 7) et un masque de sélection T $T = 3, 5, 7$). Plus élevée est la valeur de ce masque, plus grand est le nombre de pixels de bord mais plus grossière est l'approche. Dans le diagramme de flux, cette étape de sélection est représentée par "x=Edge Detection(b, T, X)". La section suivante montre comment le schéma s'adapte aux valeurs de m et de x.

9.1.2/ UN EMBARQUEMENT ADAPTATIF

Nous argumentons que le schéma d'embarquement doit s'adapter au message m et au nombre de bits disponibles pour cet embarquement. Deux stratégies sont possibles dans STABYLO. Dans la première, dite *adaptative*, le taux d'embarquement (rapport entre le nombre de bits embarqués et le nombre de pixels modifiés) dépend du nombre de bits disponibles à l'issue de l'extraction des pixels de bords. Si ce nombre de bits est inférieur

au double de la taille du message, celui-ci est découpé en plusieurs parties. Dans la seconde dite *fixe*, ce taux est fixe et l'algorithme augmente itérativement la valeur de T jusqu'à obtenir à nouveau deux fois plus de bits de bords qu'il n'y en a dans le message.

STABYLO applique alors par défaut l'algorithme STC [FJF11b] pour modifier aussi peu que possible les bits parmi ceux dont il dispose. Dans le cas où c'est la stratégie adaptative qui est choisie, le paramètre ρ de cet algorithme vaut 1 pour chacun des bits. Dans le cas contraire, la valeur de ce paramètre varie en fonction du seuil T de l'algorithme de détection de bord comme suit :

$$\rho_X = \begin{cases} 1 \text{ pour un bord défini par } T = 3, \\ 10 \text{ pour un bord défini par } T = 5, \\ 100 \text{ pour un bord défini par } T = 7. \end{cases}$$

9.1.3/ EXTRACTION DU MESSAGE

Résumée à la figure 9.1(b), l'extraction du message reproduit le processus d'embarquement dans l'ordre inverse puisque chaque étape est inversible.

9.2/ ANALYSE DE COMPLEXITÉ

Dans cette section, on justifie qualificatif « LOw cost » de STABYLO en comparant l'ordre de grandeur de son temps d'exécution avec ceux des principaux schémas existants à savoir HUGO [PFB10b], WOW [HF12] et UNIWARD [HFD14]. Chacune de ces quatre méthodes commence par calculer un carte de distorsion de l'ensemble des pixels et se termine en appliquant l'algorithme STC. Comme cette dernière étape est commune à toutes les approches, on évalue sa complexité à part. Dans tout ce qui suit, on considère une image carrée de taille $n \times n$. Les preuves de ces théorèmes sont données dans [CCG15]

Théorème [27] (Complexité d'algorithmes de stéganographie)**.** — *Le schéma HUGO a une complexité de l'ordre de* $\theta(2 \times n^2(343^2 + \ln(n)))$
— *Les schémas WOW et UNIWARD ont une complexité de l'ordre de* $\theta(6n^4 \ln(n) + n^2)$.
— *Le schéma STABYLO a une complexité dont l'ordre est* $\theta((5^3 + 4T + 1)n^2)$.

D'après [FJF11b], la complexité de STC est le l'ordre de $\theta(2^h.n)$ où h est la taille de la matrice dupliquée. Cette complexité linéaire est donc négligeable par rapport au reste.

La figure 9.2 représente graphiquement les complexités des étapes d'embarquement des schémas WOW/UNIWARD, HUGO, and STABYLO en considérant des images de la taille $n \times n$ où n varie entre 512 et 4096. L'axe des y est exprimé selon une échelle logarithmique. Cette figure illustre bien le qualificatif de « LOw cost » attribué à STABYLO.

9.3/ STÉGANALYSE DE STABYLO

Comme dans le chapitre 7, la base BOSS [PFB10a] de 10,000 images (au format RAW, de taille 512×512 en niveau de gris) a été à nouveau prise pour évaluer le schéma face

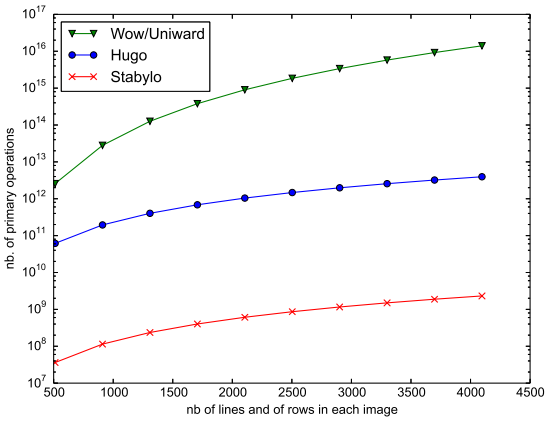


FIGURE 9.2 – Évaluation de la complexité de WOW/UNIWARD, HUGO et STABYLO

Schéma	STABYLO			HUGO		EAISLSBMR		WOW		UNIWARD	
Stratégie	fixe	adapt. (≈6.35%)		fixe	adapt.	fixe	adapt.	fixe	adapt.	fixe	adapt.
Ratio	10%	+STC(7)	+STC(6)	10%	≈6.35%	10%	≈6.35%	10%	≈6.35%	10%	≈6.35%
Ensemble Classifier	0.35	0.47	0.47	0.48	0.49	0.43	0.47	0.48	0.49	0.46	0.49

TABLE 9.1 – Stéganalyse de STABYLO.

à une épreuve de stéganalyse. Pour des rapports entre le nombre de bits embarqués et le nombre de pixels entre 1/2 et 1/9, le choix de la matrice dupliquée dans STC est celui énoncé dans les travaux de Filler [FJF11a].

Le schéma STABYLO a été systématiquement comparé à HUGO, EAISLSBMR [LHH10], WOW et UNIWARD pour les stratégies fixes (10%) et adaptatives. Pour établir la valeur de cette dernière stratégie, le filtre de Canny a été paramétré avec une valeur de $T = 3$. Lorsque b vaut 7, la taille moyenne du message pouvant être embarqué est de 16,445, *i.e.*, un taux d'embarquement moyen de 6,35%. Pour chaque image, le nombre de bits embarqué par STABYLO est mémorisé et il est demandé à chacun des autres schémas d'embarquer ce même nombre de bits.

Étant considéré comme le plus exact stéganalyseur dans le domaine spatial, Ensemble Classifier [KFH12] a été exécuté avec les caractéristiques CCPEV et SPAM [KPF10]. Les valeurs des erreurs moyennes de la phase de test sont reprises au tableau 9.1. Les schémas HUGO, WOW et UNIWARD sont moins facilement détectables que STABYLO (mais à quel prix concernant la complexité). EAILSBMR obtient des résultats semblables à STABYLO, mais encore pour une complexité plus élevée. Pour être complet, la figure 9.3 montre enfin que lorsque les taux d'embarquement sont plus élevés, STABYLO a une sécurité moindre par rapport aux quatre autres schémas.

9.4/ CONCLUSION

Le schéma STABYLO a été présenté comme une méthode efficace de stéganographie ayant des résultats comparables à HUGO, WOW et UNIWARD. pour de faibles taux d'em-

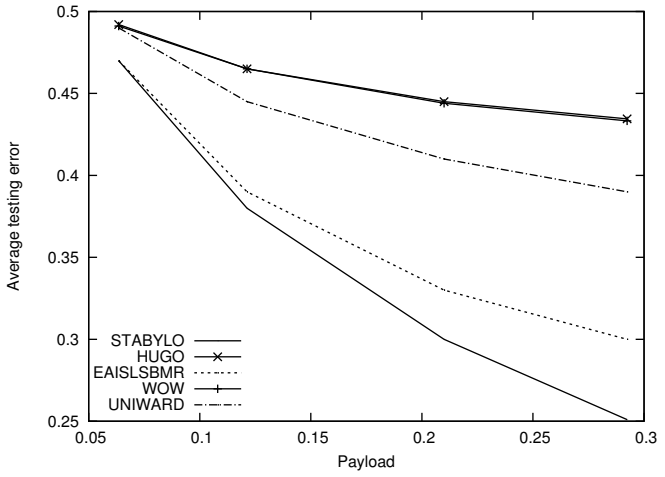


FIGURE 9.3 – Erreurs moyennes lors des tests obtenus par Ensemble Classifier

barquement. L'accent a été mis sur la complexité de l'approche pour une implantation effective, même sur des dispositifs à faible capacité de calcul.

10

SCHÉMAS DE STÉGANOGRAPHIE : LES DÉRIVÉES SECONDES

La plupart des schémas de stéganographie sont conçus de sorte à minimiser une fonction de distorsion. Dans les exemples du chapitre précédent, ces fonctions de distorsion sont construites dans l'objectif de préserver les caractéristiques de l'image. On comprend aisément que dans des régions uniformes ou sur des bords clairement définis, une modification même mineure de l'image est facilement détectable. Au contraire les textures, le bruit ou les régions chaotiques sont difficiles à modéliser. Les caractéristiques des images dont ces zones ont été modifiées sont ainsi similaires à celles des images initiales.

Ces régions sont caractérisées par des courbes de niveau très perturbées. Ce chapitre présente une nouvelle fonction de distorsion pour la stéganographie qui est basée sur les dérivées du second ordre, l'outil mathématique usuel pour les courbes de niveau.

Pour peu qu'on sache définir une fonction P qui associe à chaque pixel (x, y) sa valeur $P(x, y)$, les pixels tels que les dérivées secondes de P ont des valeurs élevées sont des bons candidats pour contenir un bit du message. Cependant, une telle fonction P n'est connue que de manière discrète, *i.e.*, en un nombre fini de points. Les dérivées premières et secondes ne peuvent donc pas être évaluées mathématiquement. Au mieux, on peut construire une fonction qui approxime les dérivées de P sur cet ensemble de pixels. Ordonner alors les pixels selon la matrice hessienne (*i.e.*, la matrice des dérivées secondes) n'est pas trivial puisque celle-ci contient de nombreuses valeurs pour un seul pixel donné.

On verra dans ce chapitre comment des approximations des dérivées premières et secondes pour des images numériques (Section 10.1) ont pu être obtenues. Deux propositions de dérivées secondes sont ensuite données et prouvées (Section 10.2 et Section 10.3). Une adaptation d'une fonction de distorsion existante est étudiée en Section 10.4 et des expériences sont présentées en Section 10.5. Ce chapitre a été publié dans [CCFG16].

10.1/ DES DÉRIVÉES DANS UNE IMAGE

Cette section rappelle d'abord les liens entre lignes de niveau, gradient et matrice hessienne puis analyse ensuite leur construction à l'aide de noyaux de la théorie du signal.

10.1.1/ MATRICE HESSIENNE

On considère qu'une image peut être assimilée à une fonction de $\mathbb{R}^+ \times \mathbb{R}^+$ dans $\mathbb{R}^+$ telle que la valeur $P(x, y)$ est associée à chaque pixel de coordonnées (x, y). Les variations d'une telle fonction en (x_0, y_0) peuvent être évaluées grâce au gradient $\nabla P(x_0, y_0) = \left(\frac{\partial P}{\partial x}(x_0, y_0), \frac{\partial P}{\partial y}(x_0, y_0)\right)$. Le vecteur gradient pointe dans la direction où la fonction a le plus fort accroissement. Des pixels ayant des valeurs voisines sont sur des lignes de niveaux qui sont orthogonales à ce vecteur.

Les variations du vecteur gradient s'expriment usuellement à l'aide de la matrice hessienne H des dérivées partielles de second ordre de P.

$$H = \begin{bmatrix} \dfrac{\partial^2 P}{\partial x^2} & \dfrac{\partial^2 P}{\partial x \partial y} \\ \dfrac{\partial^2 P}{\partial y \partial x} & \dfrac{\partial^2 P}{\partial y^2} \end{bmatrix}.$$

En un pixel (x_0, y_0), plus les valeurs de cette matrice sont éloignées de zéro, plus le gradient varie en ce point. Évaluer ce type de matrice est ainsi primordial en stéganographie. Cependant cette tâche n'est pas aussi triviale qu'elle n'y paraît puisque les images naturelles ne sont pas définies à l'aide de fonctions différentiables de $\mathbb{R}^+ \times \mathbb{R}^+$ dans $\mathbb{R}^+$. La suite montre comment obtenir des approximations de telles matrices.

10.1.2/ APPROCHES CLASSIQUES POUR ÉVALUER LE GRADIENT DANS DES IMAGES

Dans ce contexte, les approches les plus utilisées pour évaluer un gradient sont "Sobel", "Prewitt", "Différence centrale" et " Différence intermédiaire". Chacune de ces approches applique un produit de convolution $*$ entre un noyau K (rappelé dans le tableau 10.1) et une fenêtre A de taille 3×3. Le résultat $A * K$ est une approximation du gradient horizontal *i.e.*, $\dfrac{\partial P}{\partial x}$. Soit K' le résultat de la rotation d'un angle $\pi/2$ appliquée à K. La composante verticale du gradient, $\dfrac{\partial P}{\partial y}$ est obtenue de manière similaire en évaluant $A * K'$. Lorsqu'on applique ceci sur toute la matrice image, on obtient une matrice de même taille pour chacune des dérivées partielles.

Les deux éléments de la première colonne (respectivement de la seconde) de la matrice hessienne sont le résultat du calcul du gradient sur la matrice $\dfrac{\partial P}{\partial x}$ (resp. sur la matrice $\dfrac{\partial P}{\partial y}$).

10.1.3/ MATRICES HESSIENNES INDUITES PAR DES APPROCHES DE GRADIENT D'IMAGES

Il est connu que $\dfrac{\partial^2 P}{\partial x \partial y}$ est égal à $\dfrac{\partial^2 P}{\partial y \partial x}$ si les méthodes qui calculent le gradient et le gradient du gradient (la matrice hessienne) sont les mêmes. Le tableau 10.2 résume les noyaux K''_{x^2} et K''_{xy} qui permettent de calculer respectivement $\dfrac{\partial^2 P}{\partial x^2}$ et $\dfrac{\partial^2 P}{\partial x \partial y}$ comme

TABLE 10.1 – Noyaux usuels pour évaluer des gradients d'images

Nom	Sobel	Prewitt
Noyau	$Ks = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$	$Kp = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$
Nom	**Différence centrale**	**Différence Intermédiaire**
Noyau	$Kc = \begin{bmatrix} 0 & 0 & 0 \\ -\frac{1}{2} & 0 & +\frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix}$	$Ki = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

un produit de convolution pour chacun des opérateurs de gradient rappelés à la section précédente.

Le noyau Ks''_{x^2} permet de détecter si le pixel central appartient à un bord "vertical", même si celui-ci contient du bruit, en considérant ses voisins verticaux. Ces derniers sont vraiment pertinents dans un objectif de détecter les bords. Cependant, leur lien avec les lignes de niveau n'est pas direct. De plus tous les pixels qui sont dans la deuxième et la quatrième colonne de ce noyau sont ignorés. Le noyau de Prewitt a des propriétés similaires. Le noyau de différence centrale Kc''_{x^2} n'est pas influencé par les voisins verticaux du pixel central et peut paraître plus adapté ici. Cependant, le noyau Kc''_{xy} perd aussi les valeurs des pixels qui sont alignés verticalement et diagonalement avec le pixel central. Enfin, le noyau de différence intermédiaire Ki''_{x^2} décale à gauche la valeur des variations horizontales de $\dfrac{\partial P}{\partial x}$: le pixel central $(0,0)$ reçoit exactement la valeur $\dfrac{P(0,2)-P(0,1)}{1} - \dfrac{P(0,1)-P(0,0)}{1}$, qui est une approximation de $\dfrac{\partial P}{\partial x}(0,1)$ et non de $\dfrac{\partial P}{\partial x}(0,0)$. De plus, le noyau de différence intermédiaire Ki''_{xy} ne concerne que les pixels du coin supérieur droit, en perdant toutes les autres informations. La section suivante propose une autre approche pour calculer les lignes de niveau avec une précision accrue.

10.2/ DES NOYAUX POUR DES LIGNES DE NIVEAU

On ne se restreint pas aux noyaux de taille fixe (comme 3×3 or 5×5 dans les schémas précédents). Au contraire, on considère des noyaux de taille variable $(2n+1) \times (2n+1)$, $n \in \{1, 2, \ldots, N\}$, où N est un paramètre de l'approche. Les variations horizontales du

gradient sont extraites grâce au noyau de taille $(2n+1)\times(2n+1)$:

$$Ky''_{x^2} = \begin{pmatrix} 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \\ \frac{1}{2n}\ 0 \dots 0 & -\frac{2}{2n} & 0 \dots 0\ \frac{1}{2n} \\ 0 & \dots & 0 \\ \vdots & & \vdots \\ 0 & \dots & 0 \end{pmatrix}$$

Lorsque le produit de convolution est appliqué sur une fenêtre $(2n+1)\times(2n+1)$, le résultat est $\frac{1}{2}\left(\frac{P(0,n)-P(0,0)}{n} - \frac{P(0,0)-P(0,-n)}{n}\right)$, qui représente en effet les variation horizontales de la partie horizontale du gradient autour du pixel central. On obtient donc bien une approximation de $\frac{\partial^2 P}{\partial x^2}$. Lorsque n vaut 1, ce noyau est une version centrée du noyau horizontal de différence intermédiaire. Ki''_{x^2} à un facteur $1/2$ près). Lorsque n vaut 2, on retrouve Kc''_{x^2}.

Les variations verticales du gradient sont aussi obtenues en faisant subir à Ky''_{x^2} une rotation d'angle $\pi/2$. Les variations diagonales sont obtenues à l'aide du gradient Ky''_{xy} défini par :

$$Ky''_{xy} = \frac{1}{4}\begin{pmatrix} \frac{1}{n^2} & \dots & \frac{1}{2n} & \frac{1}{n} & 0 & -\frac{1}{n} & -\frac{1}{2n} & \dots & -\frac{1}{n^2} \\ \vdots & 0 & & \dots & & & 0 & & \vdots \\ \frac{1}{2n} & 0 & & \dots & & & 0 & & -\frac{1}{2n} \\ \frac{1}{n} & 0 & & \dots & & & 0 & & -\frac{1}{n} \\ 0 & & & \dots & & & & & 0 \\ -\frac{1}{n} & 0 & & \dots & & & 0 & & \frac{1}{n} \\ -\frac{1}{2n} & 0 & & \dots & & & 0 & & \frac{1}{2n} \\ \vdots & 0 & & \dots & & & 0 & & \vdots \\ -\frac{1}{n^2} & \dots & -\frac{1}{2n} & -\frac{1}{n} & 0 & \frac{1}{n} & \frac{1}{2n} & \dots & \frac{1}{n^2} \end{pmatrix}.$$

En effet, lorsque n vaut 1, Ky''_{xy} se retrouve en calculant la moyenne des variations horizontales de la composante verticale du gradient calculé à l'aide de Ky'_y. Pour cette valeur de n, on a $Ky''_{xy} = Kc''_{xy}$. Pour chaque nombre n, $1 < n \leq N$, Ky''_{xy} se retrouve de la même manière, c'est-à-dire en effectuant des moyennes de variations. Une preuve de la construction se trouve dans l'article [CCFG16].

L'objectif est de détecter les grandes variations des dérivées premières. Ainsi les dérivées secondes seront approximées comme les maximums des matrices hessiennes obtenues lorsque n varie entre 1 et N.

La dérivée partielle $\frac{\partial^2 P}{\partial x^2}$ est définie par

$$\frac{\partial^2 P}{\partial x^2} = \max\left\{\left|\frac{\partial^2 P}{\partial x^2}_1\right|, \ldots, \left|\frac{\partial^2 P}{\partial x^2}_N\right|\right\}. \tag{10.1}$$

où $\frac{\partial^2 P}{\partial x^2}_n$ est le résultat de l'application du noyau Ky''_{x^2} de taille $(2n+1)\times(2n+1)$. La même approche itérative est appliquée pour construire les approximations de $\frac{\partial^2 P}{\partial y \partial x}$ et de $\frac{\partial^2 P}{\partial y^2}$.

La section suivante étudie la pertinence d'interpoler une image par un polynôme lorsqu'on cherche à obtenir ces dérivées secondes.

10.3/ INTERPOLATION POLYNOMIALE POUR LE CALCUL DE LA MATRICE HESSIENNE

Soit $P(x,y)$ la valeur du pixel (x,y) et soit n, $1 \leq n \leq N$, tel que l'objectif est de trouver un polynôme d'interpolation dans la fenêtre de taille $(2n+1)\times(2n+1)$ dont le pixel central a pour indice $(0,0)$. Il existe un unique polynôme $L : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$ de degré $(2n+1)\times(2n+1)$ tel que $L(x,y) = P(x,y)$ pour chaque pixel (x,y) de cette fenêtre et ce polynôme est défini par

$$L(x,y) = \sum_{i=-n}^{n}\sum_{j=-n}^{n} P(i,j)\left(\prod_{\substack{-n\leq j'\leq n\\ j'\neq j}} \frac{x-j'}{i-j'}\right)\left(\prod_{\substack{-n\leq i'\leq n\\ i'\neq i}} \frac{x-i'}{i-i'}\right) \tag{10.2}$$

On peut facilement prouver que la dérivée partielle de L selon x est

$$\frac{\partial L}{\partial x} = \sum_{i=-n}^{n}\sum_{j=-n}^{n} P(i,j)\left(\prod_{\substack{-n\leq j'\leq n\\ j'\neq j}} \frac{y-j'}{j-j'}\right)\left(\sum_{\substack{-n\leq i'\leq n\\ i'\neq i}} \frac{1}{i-i'}\prod_{\substack{-n\leq i''\leq n\\ i''\neq i,i'}} \frac{x-i''}{i-i''}\right) \tag{10.3}$$

et ainsi en déduire que les dérivées partielles de second ordre sont

$$\frac{\partial^2 L}{\partial x^2} = \sum_{i=-n}^{n}\sum_{j=-n}^{n} P(i,j)\left(\prod_{\substack{-n\leq j'\leq n\\ j'\neq j}} \frac{y-j'}{j-j'}\right)\left(\sum_{\substack{-n\leq i'\leq n\\ i'\neq i}} \frac{1}{i-i'}\sum_{\substack{-n\leq i''\leq n\\ i''\neq i,i'}} \frac{1}{i-i''}\prod_{\substack{-n\leq i'''\leq n\\ i'''\neq i,i',i''}} \frac{x-i'''}{i-i'''}\right) \tag{10.4}$$

$$\frac{\partial^2 L}{\partial y \partial x} = \sum_{i=-n}^{n} P(i,j)\left(\sum_{\substack{-n\leq j'\leq n\\ j'\neq j}} \frac{1}{j-j'}\prod_{\substack{-n\leq j''\leq n\\ j''\neq j,j'}} \frac{y-j''}{j-j''}\right)\left(\sum_{\substack{-n\leq i'\leq n\\ i'\neq i}} \frac{1}{i-i'}\prod_{\substack{-n\leq i''\leq n\\ i''\neq i,i'}} \frac{x-i''}{i-i''}\right) \tag{10.5}$$

Ces dérivées secondes sont calculées pour chaque pixel central, *i.e.* le pixel dont l'indice est $(0,0)$ dans la fenêtre. En considérant cette particularisation, l'équation (10.4) se simplifie en

$$\frac{\partial^2 L}{\partial x^2} = \sum_{i=-n}^{n} P(i,0)\left(\sum_{\substack{-n\leq i'<i''\leq n\\ i',i''\neq i}} \frac{2}{(i-i')(i-i'')}\prod_{\substack{-n\leq i'''\leq n\\ i'''\neq i,i',i''}} \frac{i'''}{i'''-i}\right). \tag{10.6}$$

Cette dérivée partielle peut s'écrire comme un produit de convolution avec un noyau noté Ko''_{x^2}. Des instances de tels noyaux, pour $n = 2$, 3 et 4 sont données au tableau 10.3. De manière similaire, le tableau 10.4 donne deux exemples pour $n = 1$ et $n = 2$ de noyaux Ko''_{xy} permettant de calculer directement les dérivées de second ordre selon x et y en $(0,0)$. On remarque que pour $n = 1$, le noyau est égal à Kc''_{xy}.

10.4/ FONCTION DE DISTORSION

Une fonction de distorsion associe à chaque pixel (i, j) le coût ρ_{ij} de modification par ± 1. L'objectif est d'associer une valeur faible aux pixels dont toutes les dérivées secondes sont éloignées de 0 et une valeur rédhibitoire sinon. Dans WOW comme dans UNIWARD la fonction de distorsion est définie par

$$\rho_{ij}^{w} = \left(\left|\xi_{ij}^{h}\right|^{p} + \left|\xi_{ij}^{v}\right|^{p} + \left|\xi_{ij}^{d}\right|^{p} \right)^{-\frac{1}{p}}$$

où p est un nombre négatif et ξ_{ij}^{h} (resp. ξ_{ij}^{v} et ξ_{ij}^{d}) représente la pertinence horizontale (resp. verticale et diagonale) de modification. Une faible pertinence dans une direction signifie que l'embarquement dans ce pixel est inapproprié. La fonction de distorsion que l'on a retenu est une particularisation ($p = -1$) de cette dernière :

$$\rho_{ij} = \left(\left|\frac{\partial^2 P}{\partial x^2}(i,j)\right|^{-1} + \left|\frac{\partial^2 P}{\partial y^2}(i,j)\right|^{-1} + \left|\frac{\partial^2 P}{\partial y \partial x}(i,j)\right|^{-1} \right)$$

10.5/ EXPÉRIMENTATIONS

Tout d'abord, l'ensemble du code est accessible en ligne [1]. La Figure 10.1 représente les résultats d'embarquement de données dans l'image 38 du challenge BOSS [PFB10a] en suivant les deux schémas basés sur les dérivées secondes présentés dans ce chapitre. Le taux d'embarquement α est fixé à 0.4 bits par pixel et les noyaux sont construits avec $N = 4$. On remarque bien que les pixels dans les zones uniformes et les pixels dans les bords bien définis ne sont pas modifiés par l'approche tandis qu'au contraire les zones peu prévisibles (le monument par exemple) concentrent les changements.

Schéma	Image. Stego.	Différence avec le support
Approche à base de Ky		
Approche à base de Ko		

FIGURE 10.1 – Exemple de changements dus à un embarquement avec $\alpha = 0.4$

1. https://github.com/stego-content/SOS

10.5.1/ CHOIX DES PARAMÈTRES

Les deux méthodes présentées ici dépendent de noyaux dont la taille va jusqu'à $(2N + 1) \times (2N + 1)$. Cette section montre comment évaluer N pour maximiser le niveau de sécurité. Pour chaque approche ($N = 2$, 4, 6, 8, 10, 12 et 14), 1000 images stégos du challenge BOSS ont été selectionnées. La sécurité de l'approche a été évaluée avec le stéganalyseur Ensemble Classifier [KFH12]. Pour un taux d'embarquement α égal soit à 0.1 ou soit à 0.4, l'erreur moyenne de test (exprimée en pourcentage) a été calculée. Le tableau 10.5 synthétise les résultats. On observe que la taille $N = 4$ (respectivement $N = 12$) permet d'obtenir des erreurs suffisamment élevées pour l'approche basée sur Ky (resp. pour celle basée sur Ko). Ces deux valeurs de paramètres sont retenues par la suite.

10.5.2/ ÉVALUATION DE LA SÉCURITÉ

Comme dans ce qui précède, la base du challenge BOSS a été retenue. Ici c'est cependant l'ensemble des 10000 images qui a été utilisé pour évaluer la sécurité. C'est aussi les caractéristiques SRM et Ensemble Classifier qui ont été utilisées pour évaluer la sécurité de l'approche. Quatre taux d'embarquement 0.1, 0.2, 0.3 et 0.4 ont été retenus. Pour chaque expérience, l'aire sous la courbe de ROC (AUC), l'erreur moyenne de test (ATE), l'erreur OOB (OOB) sont données et tous les résultats sont synthétisés dans le tableau 10.6. Même si la sécurité est souvent plus faible que celle observée pour les outils les plus récents, les résultats concernant K_y sont encourageants car ils ne sont pas éloignés de ceux de l'état de l'art sans aucune optimisation. Enfin la faible sécurité de K_o s'explique par le fait que le polynôme interpole exactement l'image en tous les points de la fenêtre, mais il ne tient pas forcément compte des variations dans celle-ci. Les dérivées secondes sont certes faciles à exprimer, mais elles ne représentent pas nécessairement fidèlement celles de l'image.

10.6/ CONCLUSION

La principale contribution de ce chapitre est de proposer des fonctions de distorsion basées sur des approximations de dérivées secondes, l'idée sous-jacente étant qu'une zone où les lignes de niveau ne sont pas clairement définies est peu prévisible. Deux approches d'approximation ont été présentées. La première basée sur un produit de convolution, exploite des noyaux déjà intégrés dans des algorithmes de détection de bords. La seconde s'appuie sur une interpolation polynomiale de l'image. Ces deux méthodes ont été complètement implantées et leur sécurité face à des stéganalyseurs a été étudiée. Les résultats encouragent à poursuivre dans cette direction.

TABLE 10.2 – Noyaux usuels pour évaluer des gradients de second ordre d'images

Sobel	Prewitt
$Ks''_{x^2} = \begin{bmatrix} 1 & 0 & -2 & 0 & 1 \\ 4 & 0 & -8 & 0 & 4 \\ 6 & 0 & -12 & 0 & 6 \\ 4 & 0 & -8 & 0 & 4 \\ 1 & 0 & -2 & 0 & 1 \end{bmatrix}$	$Kp''_{x^2} = \begin{bmatrix} 1 & 0 & -2 & 0 & 1 \\ 2 & 0 & -4 & 0 & 2 \\ 3 & 0 & -6 & 0 & 3 \\ 2 & 0 & -4 & 0 & 2 \\ 1 & 0 & -2 & 0 & 1 \end{bmatrix}$
$Ks''_{xy} = \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -4 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & -4 & -2 \\ 1 & 2 & 0 & -2 & -1 \end{bmatrix}$	$Kp''_{xy} = \begin{bmatrix} -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 \end{bmatrix}$

Différence centrale	Différence intermédiaire
$Kc''_{x^2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & -\frac{1}{2} & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	$Ki''_{x^2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
$Kc''_{xy} = \begin{bmatrix} -\frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 \\ \frac{1}{4} & 0 & -\frac{1}{4} \end{bmatrix}$	$Ki''_{xy} = \begin{bmatrix} 0 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$

TABLE 10.3 – Noyaux Ko''_{x^2} pour calculer des dérivées de second ordre à partir d'interpolation polynomiale

n	Ko''_{x^2}
2	$\left[\frac{-1}{12}, \frac{4}{3}, \frac{-5}{2}, \frac{4}{3}\frac{-1}{12}\right]$
3	$\left[\frac{1}{90}, \frac{-3}{20}, \frac{3}{2}, \frac{-49}{18}, \frac{3}{2}, \frac{-3}{20}, \frac{1}{90}\right]$
4	$\left[\frac{-1}{560}, \frac{8}{315}, \frac{-1}{5}, \frac{8}{5}, \frac{-205}{72}, \frac{8}{5}, \frac{-1}{5}, \frac{8}{315}, \frac{-1}{560}\right]$

TABLE 10.4 – Noyaux pour les dérivées secondes en x et y lors de l'interpolation polynomiale

n	Ko''_{xy}
2	$\begin{bmatrix} \frac{1}{4} & 0 & \frac{-1}{4} \\ 0 & 0 & 0 \\ \frac{-1}{4} & 0 & \frac{1}{4} \end{bmatrix}$
3	$\begin{bmatrix} \frac{1}{144} & \frac{-1}{18} & 0 & \frac{1}{18} & \frac{-1}{144} \\ \frac{-1}{18} & \frac{4}{9} & 0 & \frac{-4}{9} & \frac{1}{18} \\ 0 & 0 & 0 & 0 & 0 \\ \frac{1}{18} & \frac{-4}{9} & 0 & \frac{4}{9} & \frac{-1}{18} \\ \frac{-1}{144} & \frac{1}{18} & 0 & \frac{-1}{18} & \frac{1}{144} \end{bmatrix}$

TABLE 10.5 – Erreur moyenne de test en fonction de la taille du noyau

	α	N						
		2	4	6	8	10	12	14
Erreur moyenne	*0.1*	39	40.2	39.7	39.8	40.1	39.9	39.8
de test pour le noyau K_y	*0.4*	15	18.8	19.1	19.0	18.6	18.7	18.7
Erreur moyenne	*0.1*	35.2	36.6	36.7	36.6	37.1	37.2	37.2
de test pour le noyau K_o	*0.4*	5.2	6.8	7.5	7.9	8.1	8.2	7.6

TABLE 10.6 – Évaluation de la sécurité

	Taux	AUC	ATE	OOB
WOW	0.1	0.6501	0.4304	0.3974
	0.2	0.7583	0.3613	0.3169
	0.3	0.8355	0.2982	0.2488
	0.4	0.8876	0.2449	0.1978
SUNIWARD	0.1	0.6542	0.4212	0.3972
	0.2	0.7607	0.3493	0.3170
	0.3	0.8390	0.2863	0.2511
	0.4	0.8916	0.2319	0.1977
MVG	0.1	0.6340	0.4310	0.4124
	0.2	0.7271	0.3726	0.3399
	0.3	0.7962	0.3185	0.2858
	0.4	0.8486	0.2719	0.2353
HUGO	0.1	0.6967	0.3982	0.3626
	0.2	0.8012	0.3197	0.2847
	0.3	0.8720	0.2557	0.2212
	0.4	0.9517	0.1472	0.1230
Approche à base de Ky	0.1	0.7378	0.3768	0.3306
	0.2	0.8568	0.2839	0.2408
	0.3	0.9176	0.2156	0.1710
	0.4	0.9473	0.1638	0.1324
Approche à base de Ko	0.1	0.6831	0.3696	0.3450
	0.2	0.8524	0.1302	0.2408
	0.3	0.9132	0.1023	0.1045
	0.4	0.9890	0.0880	0.0570

V

CONCLUSION

11

CONCLUSION ET PERSPECTIVES

Ce travail a été guidé par la volonté de comprendre une partie des avancées théoriques et pratiques autour des systèmes discrets, de formuler de nouvelles propositions dans ce champ thématique et de les démontrer lorsque nos connaissances le permettaient. Ce travail est le fruit d'une équipe et nombreux sont ceux qui y ont pris part.

Ce chapitre en présente tout d'abord une synthèse (section 11.1). Quelques perspectives qui s'en dégagent sont ensuite esquissées (section 11.2).

11.1/ SYNTHÈSES DES CONTRIBUTIONS

Les principales contributions gravitent autour des mathématiques discrètes et plus particulièrement les itérations de systèmes dynamiques discrets.

Pour chacun des modes et des conditions de synchronisme, il existe des critères (suffisants) de convergence globale ou locale.

Nous avons formalisé le mode des *itérations mixtes* (introduit par Pr. J. M. Bahi en 2005 notamment) qui combine synchronisme et asynchronisme (chapitre 1) et leur extension les *itérations mixtes avec délais uniformes*. Nous avons pu ainsi énoncer puis démontrer des résultats établissant que pour des conditions classiques de convergence des itérations synchrones, les itérations mixtes à délai uniforme convergent aussi vers le même point fixe.

Nous avons de plus démontré (chapitre 2) qu'on peut simuler des SDDs selon tous les modes avec l'outil SPIN de *Model-Checking* pour établir formellement leur convergence (ou pas). Nous avons énoncé puis prouvé ensuite la correction et la complétude de la démarche. Des données pratiques comme la complexité et des synthèses d'expérimentation ont aussi été fournies.

Nous nous sommes ensuite intéressés à l'étude du problème dual de l'étude de divergence d'un SDD. Nous avons proposé plusieurs méthodes de construction de fonctions permettant d'obtenir des itérations chaotiques. La première non naïve est basée sur des conditions suffisantes sur le graphe d'interaction à N sommets (chapitre 3). Une seconde méthode plus efficace permet en plus de disposer d'une chaîne de Markov doublement stochastique et s'appuie sur les cycles hamiltoniens du graphe des itérations. Elle est présentée au chapitre 6. Ces méthodes ont permis d'étendre à l'infini la classe des fonctions dont les itérations sont chaotiques.

Nous avons aussi entrepris d'étudier ces itérations et plus particulièrement leur apprentissage par un réseau de neurones. Nous avons notamment pu contribuer à montrer pratiquement qu'il est très difficile (voir impossible) de les prédire à l'aide d'outils d'intelligence artificielle (chapitre 4).

Avec la production d'une grande collection de fonctions à itérations chaotiques, nous avons donc proposé de répondre à la question suivante : comment engendrer des fonctions dont les itérations vont produire des nombres simulant correctement l'aléa. En d'autres termes, quelles fonctions peuvent être embarquées dans un PRNG ? Nous avons d'abord caractérisé les fonctions dont les itérations produisent des nombres selon une distribution uniforme (chapitre 5). Pour cela il a fallu réécrire l'algorithme de génération comme une marche aléatoire dans une partie du N-cube, se ramener ensuite à une chaîne de Markov puis enfin utiliser la théorie élaborée sur ce sujet pour conclure (chapitre 6).

Parmi les fonctions retenues, celles issues de la suppression d'un cycle hamiltonien dans un N-cube ont retenu notre attention. Nous nous sommes aussi attaché à montrer l'importance de l'équilibrage du cycle hamiltonien à enlever (chapitre 6). Nous avons de plus entrepris dans ce chapitre de trouver un majorant du nombre d'itérations suffisant à l'obtention d'une distribution uniforme

Nous avons renforcé la thématique de marquage de document numérique de l'équipe AND en embarquant ces fonctions dans des outils de watermarking. Nous avons participé à la formalisation de la méthode de marquage de médias (chapitre 7) et particularisé ceci à des images numériques fournissant un nouveau contexte pour l'étude théorique et mathématique d'algorithmes de marquage. Des instances de ces algorithmes ont été présentées en sélectionnant de manière pertinente les fonctions à itérer pour garantir une robustesse élevée.

D'autre méthodes de watermarking ont été investies (mais plus dans le domaine discret), particulièrement celles basées sur la Quantization Index Modulation (QIM), méthodes étant supposées comme les plus robustes. Nos principales contributions sur ce travail ont été d'intégrer ceci à du marquage de document PDF puis de présenter ce problème comme un problème d'optimisation (chapitre 8).

Nous avons de plus conçu l'algorithme STABYLO (chapitre 9) qui est un schéma de stéganographie basé sur l'enfouissement de l'information dans les contours présents dans une image. Cet algorithme présente un bon compromis entre sécurité fournie et complexité algorithmique. Nous avons enfin proposé d'exprimer les fonctions de distorsion classiquement utilisées en stéganographie comme des méthodes de calcul de gradient ou de matrice Hessienne. Grâce à l'étude de ces matrices, nous avons proposé un nouveau schéma de stéganographie sécurisé (chapitre 10).

11.2/ QUELQUES PERSPECTIVES

Les expériences, résultats et connaissances acquises lors de ce travail conduisent vers de nouvelles perspectives présentées ci-après.

11.2.1/ AUTOUR DES PRNGS

La démarche actuelle de génération de nombres pseudo-aléatoires consiste à marcher dans une partie d'un N-cube en choisissant son chemin à l'aide d'un générateur fourni en entrée. Or ces générateurs sont tous des fonctions de $\{0,1\}^N$ dans lui-même. Cette approche semble pouvoir se réécrire comme un produit synchrone de deux automates. L'intérêt d'une telle réécriture est qu'on pourrait exploiter les résultats théoriques et pratiques déjà connus dans la communauté des automates. Nous pensons investiguer cette voie pour améliorer notre approche, s'affranchir, à terme, de tout autre générateur et améliorer la connaissance à ce sujet.

De plus, marcher dans une partie d'un N-cube est le modèle théorique que nous avons établi pour notre classe de générateurs. On a vu, via les itérations généralisées qu'on pouvait modifier plusieurs bits en une seule itération. Les premiers travaux pratiques réalisés ont montré que le nombre d'itérations suffisant pour converger vers une distribution uniforme est plus petit que celui obtenu en marchant et, plus intéressant encore, qu'il diminue à mesure que N augmente. Pour l'instant, nous n'avons pas réussi à obtenir une majoration du nombre d'itérations pour le temps d'arrêt ce que nous pourrons faire dans un avenir proche.

Il nous paraît aussi important de déployer tout le travail fait autour des PRNG sur des plates-formes physiques. On pense aux circuits logiques programmables (FPGA) ou aux circuits intégrés dédiés à une application (ASIC). Un premier travail [BCG16] a été réalisé en ce sens et a consisté à comparer, sur FPGA uniquement, les implantations existantes de PRNGs de la littérature ainsi que celles à base d'itérations unaires. Poursuivre le déploiement sur ces deux familles d'architecture, intégrer les itérations généralisées et les combiner nous est une piste de recherche que nous allons poursuivre.

11.2.2/ DES CODES DE GRAY LOCALEMENT ET GLOBALEMENT ÉQUILIBRÉS

Enfin, pour générer une fonction dont la matrice de Markov est doublement stochastique –condition nécessaire pour fournir une sortie uniformément distribuée–, nous avons proposé principalement la méthode de suppression de chemin hamiltonien dans un N-cube. Nous avons fait sauter un premier verrou en proposant une méthode déterministe à l'extension de Robinson-Cohn. Il est apparu récemment des algorithmes permettant d'obtenir des codes de Gray localement équilibrés, c.-à-d. où la longueur du plus grand nombre d'étapes entre deux changements d'un même bit est aussi petite que possible. Dans tous les cas, aucun de ces codes n'est globalement équilibré ni même presque équilibré. Cette double propriété serait cependant très intéressante aussi bien théoriquement que pratiquement pour nos générateurs. Un second verrou consistera à adapter ces algorithmes pour proposer des codes possédant les deux propriétés d'équilibrage.

11.2.3/ STÉGANALYSE PAR DEEP LEARNING

Les démarches de stéganalyse sont souvent composées de 2 étapes : caractérisation puis classification. On extrait au préalable une grande quantité des caractéristiques du média puis on utilise une méthode de classification basée sur celles-ci. La communauté voit souvent cette seconde étape comme une boite noire et se concentre sur la construction de l'ensemble des caractéristiques les plus discriminantes. Autant que nous

sachions, les méthodes algébriques de réduction de domaine (analyse par composant principaux, SVD) ont rarement été utilisées comme une étape intermédiaire entre la caractérisation et la classification. Ces méthodes ont déjà été appliquées avec succès lorsqu'elles sont combinées avec des méthodes d'apprentissage, par exemple dans de la reconnaissance faciale. Je propose d'étudier cette piste dans ce domaine.

De plus les résultats obtenus en stéganalyse à l'aide de deep learning à base de convolutions sont très prometteurs lorsque la clef qui a servi à l'embarquement est constante. Malheureusement, lorsque la clef varie, nous n'avons pas réussi à généraliser ces avancées. Les démarches les plus efficaces demeurent celles obtenues par des approches classiques à base de caractéristiques statistiques (features) d'images. Cependant, en étudiant plus finement les features, on constate que nombreuses sont celles qui sont aussi basées sur des produits de convolution. Je propose d'étudier exhaustivement ces features pour d'abord traduire en deep-learning celles qui sont des convolutions directes. Il restera ensuite à adapter l'outil de deep learning aux caractéristiques restantes ce qui est un autre challenge scientifique.

A

PREUVES SUR LES RÉSEAUX DISCRETS

A.1/ CONVERGENCE DU MODE MIXTE

Introduisons tout d'abord une relation d'ordre $\preceq$ entre les classes d'équivalences. Formellement, $\langle p \rangle \preceq \langle q \rangle$ s'il existe un chemin de longueur α ($0 \leq \alpha < |\mathcal{K}|$) entre un élément le la classe $\langle p \rangle$ vers un élément de $\langle q \rangle$.

Lemme [1]. *Il existe un processus de renommage qui affecte un nouvel identifiant aux éléments $i \in \langle p \rangle$ et $j \in \langle q \rangle$ tel que $i \leq j$ si et seulement si $\langle p \rangle \preceq \langle q \rangle$.*

Démonstration. Tout d'abord, soient $\langle p_1 \rangle, \ldots, \langle p_l \rangle$ des classes contenant respectivement les éléments $n_1, \ldots, n_l$ qui ne dépendent d'aucune autre classe. Les éléments de $\langle p_1 \rangle$ sont renommés par $1, \ldots, n_1$, les éléments de $\langle p_i \rangle$, $2 \leq i \leq l$ sont renommés par $1 + \Sigma_{k=1}^{i-1} n_k$, $\ldots$, $\Sigma_{k=1}^{i} n_k$. On considère maintenant les classes $\langle p_1 \rangle, \ldots, \langle p_{l'} \rangle$ dont les éléments ont été renommés et soit m le plus grand indice des éléments de $\langle p_1 \rangle, \ldots, \langle p_{l'} \rangle$. Soit une autre classe $\langle p \rangle$ qui dépend exclusivement d'une classe $\langle p_i \rangle$, $1 \leq i \leq l'$ et qui contient k éléments. Les éléments de $\langle p \rangle$ sont renommés par $m+1, \ldots, m+k$. Ce processus a été appliqué sur $l'+1$ classes. Il se termine puisqu'il diminue le nombre d'éléments auquel il reste à affecter un numéro.

Il reste à montrer que cette méthode de renommage vérifie la propriété énoncée dans le lemme. Cette preuve se fait par induction sur la taille l du plus grand chemin de dépendance entre les classes.

Tout d'abord, si $\langle p \rangle \preceq \langle q \rangle$ et $\langle q \rangle$ dépend immédiatement de $\langle p \rangle$, *i.e.* le chemin le plus long entre les éléments de $\langle p \rangle$ et les éléments de $\langle q \rangle$ est de longueur 1. En raison de la méthode renommage, chaque numéro d'élément $\langle q \rangle$ est plus grand que tous ceux de $\langle p \rangle$ et la preuve est établie. Soit $\langle p \rangle$ et $\langle q \rangle$ tels que le plus long chemin de dépendance entre $\langle p \rangle$ et $\langle q \rangle$ a une longueur de $l+1$. Il existe alors une classe $\langle q' \rangle$ telle que $\langle q \rangle$ dépend immédiatement de $\langle q' \rangle$ et le chemin de dépendance le plus long entre $\langle p \rangle$ et $\langle q' \rangle$ a pour longueur l. On a ainsi $\langle q' \rangle \preceq \langle q \rangle$ et pour tout k, j tels que $k \in \langle q' \rangle$ et $j \in \langle q \rangle$, $k \leq j$. Par hypothèse d'induction, $\langle p \rangle \preceq \langle q' \rangle$ et pour chaque i, k tels que $i \in \langle p \rangle$ et $k \in \langle q' \rangle$, $i \leq k$ et le résultat est établi. □

On peut remarquer que ce processus de renommage est inspiré des *graphes par couches* de Golès et Salinas [GCS08].

du théorème 5. Le reste de la preuve est fait par induction sur le numéro de classe. Considérons la première classe $\langle b_1 \rangle$ de n_1 éléments *i.e.* la classe avec le plus petit identifiant.

D'après les hypothèses du théorème, les itérations synchrones convergent vers un point fixe en un nombre fini d'itérations. Ainsi toutes les *classes sources* (indépendantes de toutes les autres classes) vont aussi converger dans le mode mixte. On peut ainsi supposer que le mode d'itération mixte avec délais uniformes fait converger les classes $\langle b_1 \rangle, \ldots, \langle b_k \rangle$ en un temps t_k. Par construction, la classe $\langle b_{k+1} \rangle$ dépend uniquement de certaines classes de $\langle b_1 \rangle, \ldots, \langle b_k \rangle$ et éventuellement d'elle-même. Il existe un nombre d'itérations suffisamment grand t_0 tel que $D^{t_0}_{p_{k+1}p_j}$ est supérieur ou égal à t_k pour chaque $p_{k+1} \in \langle b_{k+1} \rangle$ et $p_j \in \langle b_j \rangle$, $1 \leq j \leq k$.

Il ne reste donc que des itérations synchrones entre les éléments de $\langle b_{k+1} \rangle$ en démarrant dans des configurations où tous les éléments de $\langle b_j \rangle$, $1 \leq j \leq k$, ont des valeurs constantes. D'après les hypothèses du théorème, cela converge. □

A.2/ CORRECTION ET COMPLÉTUDE DE LA VÉRIFICATION DE CONVERGENCE PAR SPIN

Cette section donne les preuves des deux théorèmes de correction et complétude du chapitre 2.

Lemme 2 (Stratégie équivalente)**.** *Soit ϕ un système dynamique discret de stratégie $(S^t)^{t\in\mathbb{N}}$ et ψ sa traduction en PROMELA. Il existe une exécution de ψ sous hypothèse d'équité faible telle le scheduler met à jour les éléments of S^t donnés par* `update_elems` *à l'itération t.*

Démonstration. La preuve est directe pour $t = 0$. Supposons qu'elle est établie jusqu'en t valant un certain t_0. On considère des stratégies pseudo-périodiques. Grâce à l'hypothèse d'équité faible, `update_elems` modifie les éléments de S^t à l'itération t. □

Dans ce qui suit, soit Xd^t_{ji} la valeur de `Xd[`j`].v[`i`]` après le $t^{\text{ème}}$ appel à la fonction `fetch_values`. De plus, soit Y^k_{ij} l'élément à l'indice k dans le canal `channels[i].sent[j]` de taille m, $m \leq \delta_0$; Y^0_{ij} et Y^{m-1}_{ij} sont respectivement la tête et la queue du canal. De plus, soit $(M^t_{ij})^{t\in\{1,1.5,2,2.5,\ldots\}}$ une séquence telle que M^t_{ij} est une fonction partielle qui associe à chaque k, $0 \leq k \leq m-1$, le tuple $(Y^k_{ij}, a^k_{ij}, c^k_{ij})$ en entrant dans la fonction `update_elems` à l'itération t où Y^k_{ij} est la valeur du canal `channels[i].sent[j]` à l'indice k, a^k_{ij} est la date (antérieure à t) mémorisant quand Y^k_{ij} est ajouté et c^k_{ij} est le premier temps où cette valeur est accessible à j. La valeur est supprimée du canal $i \to j$ à la date $c^k_{ij} + 1$. M^t_{ij} a la signature suivante :

$$\begin{array}{rccl} M^t_{ij}: & \{0,\ldots,max-1\} & \to & E_i \times \mathbb{N} \times \mathbb{N} \\ & k \in \{0,\ldots,m-1\} & \mapsto & M_{ij}(k) = (Y^k_{ij}, a^k_{ij}, c^k_{ij}). \end{array}$$

Intuitivement, M^t_{ij} est la mémoire du canal `channels[i].sent[j]` à l'itération t. On note que le domaine de chaque M^1_{ij} est $\{0\}$ et $M^1_{ij}(0) =$ `(Xp[i],0,0)` : en effet le processus `init` initialise `channels[i].sent[j]` avec `Xp[i]`.

Montrons comment l'indéterminisme des deux fonctions `fetch_values` et `diffuse_values` permet de modéliser l'équation (1.5). La fonction M_{ij}^{t+1} est obtenue à l'aide de mises à jour successives de M_{ij}^{t} au travers des deux fonctions `fetch_values` and `diffuse_values`. Par abus, soit $M_{ij}^{t+1/2}$ la valeur de M_{ij}^{t} après la première fonction pendant l'itération t.

Dans ce qui suit, on considère les éléments i et j dans [N]. A l'itération t, $t \geq 1$, soit $(Y_{ij}^0, a_{ij}^0, c_{ij}^0)$ la valeur de $M_{ij}^t(0)$ en entrant dans la fonction `fetch_values`. Si t est égal à $c_{ij}^0 + 1$ alors on exécute l'instruction qui affecte Y_{ij}^0 (*i.e.*, la valeur de tête du `channels[i].sent[j]`) à Xd_{ji}^t. Dans ce cas, la fonction M_{ij}^t est mise à jour comme suit : $M_{ij}^{t+1/2}(k) = M_{ij}^t(k+1)$ pour chaque k, $0 \leq k \leq m-2$ et $m-1$ est supprimée du domaine de $M_{ij}^{t+1/2}$. Sinon, (*i.e.*, lorsque $t < c_{ij}^0 + 1$ ou lorsque le domaine de M_{ij} est vide) l'instruction `skip` est exécutée et $M_{ij}^{t+1/2} = M_{ij}^t$.

Dans la fonction `diffuse_values`, s'il existe un τ, $\tau \geq t$ tel que $D_{ji}^\tau = t$, soit alors c_{ij} défini par $\min\{l \mid D_{ji}^l = t\}$. Dans ce cas, on exécute l'instruction qui ajoute la valeur `Xp[i]` dans la queue du canal `channels[i].sent[j]`. Alors, M_{ij}^{t+1} est défini en étendant $M_{ij}^{t+1/2}$ à m de sorte que $M_{ij}^{t+1}(m)$ est $(\texttt{Xp[i]}, t, c_{ij})$. Sinon, (*i.e.*, lorsque $\forall l \,.\, l \geq t \Rightarrow D_{ji}^l \neq t$ est établie) l'instruction `skip` est exécutée et $M_{ij}^{t+1} = M_{ij}^{t+1/2}$.

Lemme [3] (Existence d'une exécution SPIN)**.** *Pour chaque séquence* $(S^t)^{t\in\mathbb{N}}$, $(D^t)^{t\in\mathbb{N}}$*, pour chaque fonction* F*, il existe une exécution SPIN telle que pour toute itération* t*,* $t \geq 1$*, et pour chaque* i *et* j *dans* [N] *on a la propriété suivante :*

Si le domaine de M_{ij}^t *n'est pas vide, alors*

$$\begin{cases} M_{ij}^1(0) & = & \left(X_i^{D_{ji}^0}, 0, 0\right) \\ si\ t \geq 2\ alors\ M_{ij}^t(0) & = & \left(X_i^{D_{ji}^c}, D_{ji}^c, c\right),\ c = \min\{l | D_{ji}^l > D_{ji}^{t-2}\} \end{cases} \tag{A.1}$$

De plus, on a :

$$\forall t' \,.\, 1 \leq t' \leq t \Rightarrow Xd_{ji}^{t'} = X_i^{D_{ji}^{t'-1}} \tag{A.2}$$

Enfin, pour chaque $k \in S^t$*, la valeur de la variable* `Xp[k]` *en sortant du processus* `update_elems` *est égale à* X_k^t *i.e.,* $F_k\left(X_1^{D_{k1}^{t-1}}, \ldots, X_{\mathsf{N}}^{D_{k\mathsf{N}}^{t-1}}\right)$ *à la fin de la* $t^{ème}$ *itération.*

Démonstration. La preuve est faite par induction sur le nombre d'itérations.

Situation initiale : Pour le premier item, par définition de M_{ij}^t, on a $M_{ij}^1(0) = (\texttt{Xp[i]}, 0, 0)$ qui est égal à $\left(X_i^{D_{ji}^0}, 0, 0\right)$. Ensuite, le premier appel à la fonction `fetch_value` soit affecte la tête de `channels[i].sent[j]` à `Xd[j].v[i]` soit ne modifie par `Xd[j].v[i]`. Grâce au processus `init` process, les deux cas sont égaux à `Xp[i]`, *i.e.*, X_i^0. L'équation (A.2) est ainsi établie.

Pour le dernier item, soit k, $0 \leq k \leq \mathsf{N}-1$. A la fin de la première exécution du processus `update_elems`, la valeur de `Xp[k]` est $F(\texttt{Xd[}k\texttt{].v[0]}, \ldots, \texttt{Xd[}k\texttt{].v[N}-1\texttt{]})$. Ainsi par définition de Xd, ceci est égal à $F(Xd_{k0}^1, \ldots, Xd_{k\mathsf{N}-1}^1)$. Grâce à l'équation (A.2), on peut conclure la preuve.

Induction : Supposons maintenant que le lemme 3 est établi jusqu'à l'itération l.

Tout d'abord, si le domaine de définition de la fonction M_{ij}^l n'est pas vide, par hypothèse d'induction $M_{ij}^l(0)$ est $\left(X_i^{D_{ji}^c}, D_{ji}^c, c\right)$ où c est $\min\{k|D_{ji}^k > D_{ji}^{l-2}\}$.

A l'itération l, si $l < c+1$ alors l'instruction `skip` est exécutée dans la fonction `fetch_values`. Ainsi, $M_{ij}^{l+1}(0)$ est égal à $M_{ij}^l(0)$. Puisque $c > l-1$, alors $D_{ji}^c > D_{ji}^{l-1}$ et donc, c est $\min\{k|D_{ji}^k > D_{ji}^{l-1}\}$. Cela implique que $D_{ji}^c > D_{ji}^{l-2}$ et $c = \min\{k|D_{ji}^k > D_{ji}^{l-2}\}$.

On considère maintenant qu'à l'itération l, celui-ci vaut $c+1$. Dit autrement, M_{ij} est modifié en fonction du domaine $dom(M_{ij}^l)$ de M_{ij}^l :

- si $dom(M_{ij}^l) = \{0\}$ et $\forall k \,.\, k \geq l \Rightarrow D_{ji}^k \neq l$ sont vraies, alors $dom(M_{ij}^{l+1})$ est vide et le premier item du lemme est vérifié ;
- si $dom(M_{ij}^l) = \{0\}$ et $\exists k \,.\, k \geq l \wedge D_{ji}^k = l$ sont vraies, alors $M_{ij}^{l+1}(0)$ vaut $(\texttt{Xp[i]}, l, c_{ij})$ qui est ajouté dans la fonction `diffuse_values` de sorte que $c_{ij} = \min\{k \mid D_{ji}^k = l\}$.
  Prouvons qu'on peut exprimer $M_{ij}^{l+1}(0)$ comme $\left(X_i^{D_{ji}^{c'}}, D_{ji}^{c'}, c'\right)$ où c' vaut $\min\{k|D_{ji}^k > D_{ji}^{l-1}\}$. Tout d'abord, il n'est pas difficile de prouver que $D_{ji}^{c_{ij}} = l \geq D_{ji}^l > D_{ji}^{l-1}$ et que $c_{ij} \geq c'$. Ensuite, comme $dom(M_{ij}^l) = \{0\}$, alors, entre les itérations $D_{ji}^c + 1$ et $l-1$, la fonction `diffuse_values` n'a pas mis à jour M_{ij}. On a ainsi la propriété
  $$\forall t, k \,.\, D_{ji}^c < t < l \wedge k \geq t \Rightarrow D_{ji}^k \neq t.$$
  En particulier, on a $D_{ji}^{c'} \notin \{D_{ji}^c + 1, \ldots, l-1\}$. On peut donc appliquer le troisième item de l'hypothèse d'induction pour déduire $\texttt{Xp[i]} = X_i^{D_{ji}^{c'}}$ et on peut conclure.
- Si $\{0,1\} \subseteq dom(M_{ij}^l)$, alors $M_{ij}^{l+1}(0)$ vaut $M_{ij}^l(1)$. Soit $M_{ij}^l(1) = \left(\texttt{Xp[i]}, a_{ij}, c_{ij}\right)$. Par construction, a_{ij} vaut $\min\{t'|t' > D_{ji}^c \wedge (\exists k \,.\, k \geq t' \wedge D_{ji}^k = t')\}$ et c_{ij} est $\min\{k|D_{ji}^k = a_{ij}\}$. Montrons que c_{ij} est égal à $\min\{k|D_{ji}^k > D_{ji}^{l-1}\}$, noté plus tard c'. On a tout d'abord $D_{ji}^{c_{ij}} = a_{ij} > D_{ji}^c$. Puisque c par définition est supérieur ou égal à $l-1$, alors $D_{ji}^{c_{ij}} > D_{ji}^{l-1}$ et donc $c_{ij} \geq c'$. Ensuite, puisque $c = l-1$, c' vaut $\min\{k|D_{ji}^k > D_{ji}^c\}$ et donc $a_{ij} \leq D_{ji}^{c'}$. Ainsi, $c_{ij} \leq c'$ et on peut conclure comme dans la partie précédente.

Le cas où le domaine $dom(M_{ij}^l)$ est vide mais où la formule $\exists k \,.\, k \geq l \wedge D_{ji}^k = l$ est vraie est équivalent au second cas ci-dessus et n'est pas présenté.

Concentrons nous sur la formule (A.2). A l'itération $l+1$, soit c' défini par $c' = \min\{k|D_{ji}^k > D_{ji}^{l-1}\}$. Deux cas peuvent apparaître selon que D_{ji}^l et D_{ji}^{l-1} sont égaux ou non.

- Si $D_{ji}^l = D_{ji}^{l-1}$, puisque $D_{ji}^{c'} > D_{ji}^{l-1}$, alors $D_{ji}^{c'} > D_{ji}^l$ et donc c' est différent de l. L'exécution de SPIN ne modifie pas Xd_{ji}^{l+1}. On a ainsi $Xd_{ji}^{l+1} = Xd_{ji}^l = X_i^{D_{ji}^{l-1}} = X_i^{D_{ji}^l}$.
- Sinon, D_{ji}^l et plus grand que D_{ji}^{l-1} et c est donc égal à l. Selon l'équation (A.1), on a $M_{ij}^{l+1}(0) = (X_i^{D_{ji}^l}, D_{ji}^l, l)$. Ainsi l'exécution SPIN affecte $X_i^{D_{ji}^l}$ à Xd_{ji}^{l+1}, ce qui termine la preuve (A.2).

Il reste à prouver la partie inductive de la troisième partie du lemme. Soit k, $k \in S^{l+1}$. A l'issue de la première exécution du processus `update_elems`, on a $\texttt{Xp[}k\texttt{]} = F(\texttt{Xd[}k\texttt{][0]}, \ldots, \texttt{Xd[}k\texttt{][}n\texttt{-1]})$+. Par définition $Xd = F(Xd_{k0}^{l+1}, \ldots, Xd_{kn-1}^{l+1})$. Grâce à (A.2) déjà prouvée, on peut conclure la preuve. □

Lemme 4. *Borner la taille du canal à max = δ_0 est suffisant lorsqu'on simule un système dynamique dont les délais sont bornés par δ_0.*

Démonstration. Pour chaque i et j, à chaque itération $t+1$, comme les délais sont bornés par δ_0, l'élément i doit connaître au plus δ_0 valeurs qui sont $X_j^t, \ldots, X_j^{t-\delta_0+1}$. Elles peuvent être mémorisées dans n'importe quel canal de taille δ_0. □

Théorème 6 (Correction de la traduction vers Promela)**.** *Soit ϕ un modèle de système dynamique discret et ψ sa traduction PROMELA. Si ψ vérifie la propriété LTL (2.1) sous hypothèse d'équité faible, alors les itérations de ϕ sont universellement convergentes.*

Démonstration. Montrons la contraposée du théorème. Le lemme précédent a montré que pour chaque séquence d'itérations du système dynamique discret, Il existe une exécution du modèle PROMELA qui la simule. Si des itérations du système dynamique discret sont divergentes, leur exécution vont empêcher le modèle PROMELA de se stabiliser, *i.e.* ce dernier ne vérifiera pas la propriété LTL (2.1). □

Théorème 7 (Complétude de la traduction vers Promela)**.** *Soit ϕ un modèle de système dynamique discret et ψ sa traduction. Si ψ ne vérifie pas la propriété LTL (2.1) sous hypothèse d'équité faible, alors les itérations de ϕ ne sont pas universellement convergentes.*

Démonstration. Pour chaque modèle ψ qui ne vérifie pas la propriété LTL (2.1), il est immédiat de construire les itérations correspondantes du système dynamique, dont la stratégie est pseudo-périodique en raison de la propriété d'équité faible..

□

B

PREUVES SUR LES SYSTÈMES CHAOTIQUES

B.1/ PREUVE QUE d EST UNE DISTANCE SUR $\mathcal{X}_g$

Pour $S, S' \in \mathcal{P}(\{1, \dots, \mathsf{N}\})$, on définit

$$d_S(S, S') = \frac{9}{\mathsf{N}} \sum_{t \in \mathbb{N}} \frac{|S_t \Delta S'_t|}{10^{t+1}}.$$

Montrons que d_S est une distance sur $\mathcal{P}(\{1, \dots, \mathsf{N}\})$ et ainsi que d définie à l'équation (3.6) est une distance.

Soit S, S' et S'' trois parties de $[\mathsf{N}]$.

— De manière évidente, $d_s(S, S')$ est positive ou bien nulle si et seulement si S et S' sont égales.

— Comme la différence symétrique est commutative, la valeur de $d_S(S, S')$ est égale à celle de $d_S(S', S)$.

— On a enfin la succession d'éléments suivants :

$$\begin{aligned}
S \Delta S' &= (S \cap \overline{S'}) \cup (\overline{S} \cap S') \\
&= (S \cap \overline{S'} \cap S'') \cup (S \cap \overline{S'} \cap \overline{S''}) \cup (\overline{S} \cap S' \cap S'') \cup (\overline{S} \cap S' \cap \overline{S''}) \\
&\subseteq (S \cap \overline{S'} \cap S'') \cup (S \cap \overline{S'} \cap \overline{S''}) \cup (\overline{S} \cap S' \cap S'') \cup (\overline{S} \cap S' \cap \overline{S''}) \cup \\
&\quad (\overline{S} \cap \overline{S'} \cap S'') \cup (S \cap S' \cap \overline{S''}) \cup (\overline{S} \cap \overline{S'} \cap S'') \cup (S \cap S' \cap \overline{S''}) \\
&= (\overline{S'} \cap S'') \cup (S \cap \overline{S''}) \cup (\overline{S} \cap S'') \cup (S' \cap \overline{S''}) \\
&= (S \Delta S'') \cup (S'' \Delta S')
\end{aligned}$$

On en déduit ainsi que $|S \Delta S'| \leq |S \Delta S''| + |S'' \Delta S'|$ et donc que l'égalité triangulaire $d_S(S, S') \leq d_S(S, S'') + d_S(S'', S')$ est établie.

B.2/ CARACTÉRISATION DES FONCTIONS f RENDANT CHAOTIQUE G_{f_g} DANS $(\mathcal{X}_g, d)$

Commençons par caractériser l'ensemble $\mathcal{T}$ des fonctions transitives dans le cas des itérations généralisées.

Théorème 12. *G_{f_g} est transitive si et seulement si* GIG(f) *est fortement connexe.*

Démonstration. $\Longleftarrow$ Supposons que $\mathrm{GIG}(f)$ soit fortement connexe. Soient (x, S) et (x', S') deux points de $\mathcal{X}_g$ et $\varepsilon > 0$. On construit la stratégie $\tilde{S}$ telle que la distance entre $(x, \tilde{S})$ et (x, S) est inférieure à ε et telle que les itérations parallèles de G_{f_g} depuis $(x, \tilde{S})$ mènent au point (x', S').

Pour cela, on pose $t_1 = -\lfloor \log_{10}(\varepsilon) \rfloor$ et x'' la configuration de $\mathbb{B}^{\mathbb{N}}$ obtenue depuis (x, S) après t_1 itérations parallèles de G_{f_g}. Comme $\mathrm{GIG}(f)$ est fortement connexe, il existe une stratégie S'' et un entier t_2 tels que x' est atteint depuis (x'', S'') après t_2 itérations de G_{f_g}.

Considérons à présent la stratégie $\tilde{S} = (s_0, \dots, s_{t_1-1}, s''_0, \dots, s''_{t_2-1}, s'_0, s'_1, s'_2, s'_3 \dots)$. Il est évident que (x', s') est atteint depuis $(x, \tilde{S})$ après $t_1 + t_2$ itérations parallèles de G_{f_g}. Puisque $\tilde{s}_t = s_t$ pour $t < t_1$, grâce au choix de t_1, on a $d((x, S), (x, \tilde{S})) < \epsilon$. Par conséquent, G_{f_g} est transitive.

$\Longrightarrow$ Démontrons la contraposée. Si $\mathrm{GIG}(f)$ n'est pas fortement connexe, alors il existe deux configurations x et x' telles qu'aucun chemin de $\mathrm{GIG}(f)$ ne mène de x à x'. Soient S et S' deux stratégies et $\varepsilon \in]0; 1[$. Alors, pour tout (x'', S'') tel que $d((x'', S''), (x, S)) < \varepsilon$ on a x'' qui est égal à x. Comme il n'existe aucun chemin de $\mathrm{GIG}(f)$ qui mène de x à x', les itérations de G_{f_g} à partir de $(x'', S'') = (x, S'')$ ne peuvent atteindre que des points (x''', S''') de $\mathcal{X}_g$ tels que $x''' \neq x'$, et donc ne peuvent pas atteindre (x', S'). On peut remarquer que, du fait que $x''' \neq x'$, elles n'atteignent que des points de $\mathcal{X}_g$ dont la distance à (x', S') est supérieure à 1. Pour tout entier naturel t, on a $G^t_{f_g}(x'', S'') \neq (x', S')$. Ainsi G_{f_g} n'est pas transitive et par contraposée, on a la démonstration souhaitée. □

Prouvons à présent le théorème suivant :

Théorème 13. $\mathcal{T} \subset \mathcal{R}$.

Démonstration. Soit $f : \mathbb{B}^{\mathbb{N}} \to \mathbb{B}^{\mathbb{N}}$ telle que G_{f_g} est transitive (*i.e.* f appartient à $\mathcal{T}$). Soit $(x, S) \in \mathcal{X}_g$ et $\varepsilon > 0$. Pour prouver que f appartient à $\mathcal{R}$, il suffit de prouver qu'il existe une stratégie $\tilde{S}$ telle que la distance entre $(x, \tilde{S})$ et (x, S) est inférieure à ε et telle que $(x, \tilde{S})$ est un point périodique.

Soit $t_1 = -\lfloor \log_{10}(\varepsilon) \rfloor$ et soit x' la configuration obtenue après t_1 itérations de G_{f_g} depuis (x, S). D'après la proposition précédente, $\mathrm{GIG}(f)$ est fortement connexe. Ainsi, il existe une stratégie S' et un nombre $t_2 \in \mathbb{N}$ tels que x est atteint depuis (x', S') après t_2 itérations de G_{f_g}.

Soit alors la stratégie $\tilde{S}$ qui alterne les t_1 premiers termes de S avec les t_2 premiers termes de S'. Ainsi $\tilde{S}$ est définie par

$$(s_0, \dots, s_{t_1-1}, s'_0, \dots, s'_{t_2-1}, s_0, \dots, s_{t_1-1}, s'_0, \dots, s'_{t_2-1}, s_0, \dots).$$

Il est évident que $(x, \tilde{S})$ s'obtient à partir de $(x, \tilde{S})$ après $t_1 + t_2$ itérations parallèles de G_{f_g}. Ainsi $(x, \tilde{S})$ est un point périodique. Puisque $\tilde{s}_t$ est égal à s_t pour $t < t_1$, d'après le choix de t_1, on a $d((x, S), (x, \tilde{S})) < \epsilon$. □

On peut conclure que $\mathcal{C} = \mathcal{R} \cap \mathcal{T} = \mathcal{T}$.

B.3/ CONDITIONS SUFFISANTES POUR UN GIU(f) FORTEMENT CONNEXE

Soit $\alpha \in \mathbb{B}$. On nomme f^α la fonction de $\mathbb{B}^{N-1}$ dans lui-même définie pour chaque $x \in \mathbb{B}^{N-1}$ par

$$f^\alpha(x) = (f_1(x,\alpha), \ldots, f_{N-1}(x,\alpha)).$$

On nomme GIU(f)$^\alpha$ le sous-graphe de GIU(f) engendré par le sous-ensemble $\mathbb{B}^{N-1} \times \{\alpha\}$ de $\mathbb{B}^N$.

Énonçons et prouvons tout d'abord les lemmes techniques suivants :

Lemme 5. *$G(f^\alpha)$ est un sous-graphe de $G(f)$: chaque arc de $G(f^\alpha)$ est un arc de $G(f)$. De plus si $G(f)$ n'a pas d'arc de* N *vers un autre sommet $i \neq$ N, alors on déduit $G(f^\alpha)$ de $G(f)$ en supprimant le sommet* N *ainsi que tous les arcs dont* N *est soit l'extrémité, soit l'origine (et dans ce dernier cas, les arcs sont des boucles sur* N*).*

Démonstration. Supposons que $G(f^\alpha)$ possède un arc de j vers i de signe s. Par définition, il existe un sommet $x \in \mathbb{B}^{N-1}$ tel que $f^\alpha_{ij}(x) = s$, et puisque $f^\alpha_{ij}(x) = f_{ij}(x,\alpha)$, on en déduit que $G(f)$ possède un arc de j à i de signe s. Ceci prouve la première assertion. Pour démontrer la seconde, il suffit de prouver que si $G(f)$ a un arc de j vers i de signe s, avec $i, j \neq$ N, alors $G(f^\alpha)$ contient aussi cet arc. Ainsi, supposons que $G(f)$ a un arc de j vers i de signe s, avec $i, j \neq$ N. Alors, il existe $x \in \mathbb{B}^{N-1}$ et $\beta \in \mathbb{B}$ tels que $f_{ij}(x,\beta) = s$. Si $f_{ij}(x,\beta) \neq f_{ij}(x,\alpha)$, alors f_i dépend du N$^{\text{ème}}$ composant, ce qui est en contradiction avec les hypothèses. Ainsi $f_{ij}(x,\alpha)$ est égal à s. On a donc aussi $f^\alpha_{ij}(x) = s$. Ainsi $G(f^\alpha)$ possède un arc arc de j vers i de signe s. □

Lemme 6. *Les graphes* GIU(f^α) *et* GIU(f)$^\alpha$ *sont isomorphes.*

Démonstration. Soit h la bijection de $\mathbb{B}^{N-1}$ vers $\mathbb{B}^{N-1} \times \{\alpha\}$ définie par $h(x) = (x,\alpha)$ pour chaque $x \in \mathbb{B}^{N-1}$. On voit facilement que h permet de définir un isomorphisme entre GIU(f^α) et GIU(f)$^\alpha$: GIU(f^α) possède un arc de x vers y si et seulement si GIU(f)$^\alpha$ a un arc de $h(x)$ vers $h(y)$. □

On peut alors prouver le théorème :

Théorème 15. *Soit f une fonction de $\mathbb{B}^N$ vers lui-même telle que :*

1. *$\Gamma(f)$ n'a pas de cycle de longueur supérieure ou égale à deux ;*
2. *chaque sommet de $\Gamma(f)$ qui possède une boucle positive a aussi une boucle négative ;*
3. *chaque sommet de $\Gamma(f)$ est accessible depuis un sommet qui possède une boucle négative.*

Alors, GIU(f) *est fortement connexe.*

Démonstration. La preuve se fait par induction sur N. Soit f une fonction de $\mathbb{B}^N$ dans lui-même et qui vérifie les hypothèses du théorème. Si N $= 1$ la démonstration est élémentaire : en raison du troisième point du théorème, $G(f)$ a une boucle négative ; ainsi $f(x) = \overline{x}$ et GIU(f) est un cycle de longueur 2. On suppose donc que N > 1 et que le théorème est valide pour toutes les fonctions de $\mathbb{B}^{N-1}$ dans lui-même. En raison du

premier point du théorème, $G(f)$ contient au moins un sommet i tel qu'il n'existe pas dans $G(f)$ d'arc de i vers un autre sommet $j \neq i$. Sans perte de généralité, on peut considérer que ce sommet est N. Alors, d'après le lemme 5, f^0 et f^1 vérifient les conditions de l'hypothèse. Alors, par hypothèse d'induction $\text{GIU}(f^0)$ et $\text{GIU}(f^1)$ sont fortement connexes. Ainsi, d'après le lemme 6, $\text{GIU}(f)^0$ et $\text{GIU}(f)^1$ sont fortement connexes. Pour prouver que $\text{GIU}(f)$ est fortement connexe, il suffit de prouver que $\text{GIU}(f)$ contient un arc $x \to y$ avec $x_{\mathsf{N}} = 0 < y_{\mathsf{N}}$ et un arc $x \to y$ avec $x_{\mathsf{N}} = 1 > y_{\mathsf{N}}$. En d'autres mots, il suffit de prouver que :

$$\forall \alpha \in \mathbb{B},\ \exists x \in \mathbb{B}^{\mathsf{N}}, \qquad x_{\mathsf{N}} = \alpha \neq f_{\mathsf{N}}(x). \tag{$*$}$$

On suppose tout d'abord que N a une boucle négative. Alors, d'après la définition de $G(f)$, il existe $x \in \mathbb{B}^{\mathsf{N}}$ tel que $f_{\mathsf{N}}(x) < 0$. Ainsi si $x_{\mathsf{N}} = 0$, on a $f_{\mathsf{N}}(x) > f_{\mathsf{N}}(\overline{x}^{\mathsf{N}})$, et donc $x_{\mathsf{N}} = 0 \neq f_{\mathsf{N}}(x)$ et $\overline{x}^{\mathsf{N}}_{\mathsf{N}} = 1 \neq f_{\mathsf{N}}(\overline{x}^{\mathsf{N}})$; et si $x_{\mathsf{N}} = 1$, on a $f_{\mathsf{N}}(x) < f_{\mathsf{N}}(\overline{x}^{\mathsf{N}})$, donc $x_{\mathsf{N}} = 1 \neq f_{\mathsf{N}}(x)$ et $\overline{x}^{\mathsf{N}}_{\mathsf{N}} = 0 \neq f_{\mathsf{N}}(\overline{x}^{\mathsf{N}})$. Dans les deux cas, la condition $(*)$ est établie.

Supposons maintenant que N n'a pas de boucle négative. D'après la seconde hypothèse, N n'a pas de boucle, *i.e.*, la valeur de $f_{\mathsf{N}}(x)$ ne dépend pas de la valeur de x_{N}. D'après la troisième hypothèse, il existe $i \in [\mathsf{N}]$ tel que $G(f)$ a un arc de i vers N. Ainsi, il existe $x \in \mathbb{B}^{\mathsf{N}}$ tel que $f_{\mathsf{N}i}(x) \neq 0$ et donc f_{N} n'est pas constante. Ainsi, il existe $x, y \in \mathbb{B}^{\mathsf{N}}$ tel que $f_{\mathsf{N}}(x) = 1$ et $f_{\mathsf{N}}(y) = 0$. Soit $x' = (x_1, \ldots, x_{\mathsf{N}-1}, 0)$ et $y' = (y_1, \ldots, y_{\mathsf{N}-1}, 1)$. Puisque la valeur de $f_{\mathsf{N}}(x)$ (resp. de $f_{\mathsf{N}}(y)$) ne dépend pas de la valeur de x_{N} (resp. de y_{N}), on a $f_{\mathsf{N}}(x') = f_{\mathsf{N}}(x) = 1 \neq x'_{\mathsf{N}}$ (resp. $f_{\mathsf{N}}(y') = f_{\mathsf{N}}(y) = 0 \neq y'_{\mathsf{N}}$). Ainsi la condition $(*)$ est établie, et le théorème est prouvé. □

C

PREUVES SUR LES GÉNÉRATEURS DE NOMBRES PSEUDO-ALÉATOIRES

C.1/ CHAÎNES DE MARKOV ASSOCIÉES À GIU(f)

Considérons le lemme technique suivant :

Lemme 7. *Soit $f : \mathbb{B}^n \to \mathbb{B}^n$, GIU($f$) son graphe d'itérations, $\check{M}$ la matrice d'adjacence de GIU(f), et M la matrice $2^n \times 2^n$ définie par $M = \frac{1}{n}\check{M}$. Alors M est une matrice stochastique régulière si et seulement si GIU(f) est fortement connexe.*

Démonstration. On remarque tout d'abord que M est une matrice stochastique par construction. Supposons M régulière. Il existe donc k tel que $M^k_{ij} > 0$ pour chaque $i, j \in [\![1; 2^n]\!]$. L'inégalité $\check{M}^k_{ij} > 0$ est alors établie. Puisque $\check{M}^k_{ij}$ est le nombre de chemins de i à j de longueur k dans GIU(f) et puisque ce nombre est positif, alors GIU(f) est fortement connexe.

Réciproquement si GIU(f) est fortement connexe, alors pour tous les sommets i et j, un chemin peut être construit pour atteindre j depuis i en au plus 2^n étapes. Il existe donc $k_{ij} \in [\![1, 2^n]\!]$ tels que $\check{M}^{k_{ij}}_{ij} > 0$. Comme tous les multiples $l \times k_{ij}$ de k_{ij} sont tels que $\check{M}^{l\times k_{ij}}_{ij} > 0$, on peut conclure que, si k est le plus petit multiple commun de $\{k_{ij}/i, j \in [\![1, 2^n]\!]\}$ alors $\forall i, j \in [\![1, 2^n]\!], \check{M}^k_{ij} > 0$. Ainsi, $\check{M}$ et donc M sont régulières. □

Ces résultats permettent formuler et de prouver le théorème annoncé.

Théorème 17 (Uniformité de la sortie de l'algorithme 1)**.** *Soit $f : \mathbb{B}^N \to \mathbb{B}^N$, GIU($f$) son graphe d'itérations , $\check{M}$ sa matrice d'adjacence et M une matrice $2^n \times 2^n$ définie par $M = \frac{1}{N}\check{M}$. Si GIU(f) est fortement connexe, alors la sortie du générateur de nombres pseudo-aléatoires détaillé par l'algorithme 1 suit une loi qui tend vers la distribution uniforme si et seulement si M est une matrice doublement stochastique.*

Démonstration. M est une matrice stochastique régulière (Lemme 7) qui a un unique vecteur de probabilités stationnaire (Théorème 16). Soit π défini par $\pi = \left(\frac{1}{2^n}, \ldots, \frac{1}{2^n}\right)$. On a $\pi M = \pi$ si et seulement si la somme des valeurs de chaque colonne de M est 1, *i.e.* si et seulement si M est doublement stochastique. □

C.2/ CHAOTICITÉ DE LA FONCTION $G_{f_u,\mathcal{P}}$ DANS $(\mathcal{X}_{\mathsf{N},\mathcal{P}}, d)$

Montrons le théorème

Théorème 18 (Une distance dans $\mathcal{X}_{\mathsf{N},\mathcal{P}}$)**.** *$d$ est une distance sur $\mathcal{X}_{\mathsf{N},\mathcal{P}}$.*

Démonstration. $d_{\mathbb{B}^{\mathsf{N}}}$ est la distance de Hamming. Prouvons que $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}$ est aussi une distance ; d sera ainsi une distance comme somme de deux distances.

- De manière évidente, $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s,\check{s}) \geqslant 0$, et si $s = \check{s}$, alors $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s,\check{s}) = 0$. Réciproquement si $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s,\check{s}) = 0$, alors $\forall k \in \mathbb{N}, v^k = \check{v}^k$ d'après la définition de d. Or les éléments entre les positions $p+1$ et $p+n$ sont nulles et correspondent à $|u^0 - \check{u}^0|$, on peut conclure que $u^0 = \check{u}^0$. On peut étendre ce résultat aux $n \times \max(\mathcal{P})$ premiers blocs engendrant $u^i = \check{u}^i$, $\forall i \leqslant v^0 = \check{v}^0$, et en vérifiant tous les $n \times \max(\mathcal{P})$ blocs, $u = \check{u}$.
- $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}$ est évidemment symétrique ($d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(s,\check{s}) = d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}(\check{s},s)$).
- l'inégalité triangulaire est établie puisque la valeur absolue la vérifie aussi.

□

Montrons que :

Lemme 8. *Le graphe d'itérations* $\mathrm{GIU}_{\mathcal{P}}(f)$ *est fortement connexe si et seulement si la fonction $G_{f_u,\mathcal{P}}$ est topologiquement transitive sur $(\mathcal{X}_{\mathsf{N},\mathcal{P}}, d)$.*

Démonstration. Supposons tout d'abord que $G_{f_u,\mathcal{P}}$ fortement connexe. Soit $x = (e,(u,v)), \check{x} = (\check{e},(\check{u},\check{v})) \in \mathcal{X}_{\mathsf{N},\mathcal{P}}$ et $\varepsilon > 0$. On cherche un point y dans une boule ouverte $\mathcal{B}(x,\varepsilon)$ et un nombre $n_0 \in \mathbb{N}$ tels que $G^{n_0}_{f_u,\mathcal{P}}(y) = \check{x}$: Cette transitivité forte entraînera la propriété de transitivité classique. On peut supposer que $\varepsilon < 1$ sans perte de généralité.

Soit $(E,(U,V))$ les éléments de y. Comme y doit appartenir à $\mathcal{B}(x,\varepsilon)$ et $\varepsilon < 1$, E est égal à e. Soit $k = \lfloor \log_{10}(\varepsilon) \rfloor + 1$. La distance $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}((u,v),(U,V))$ est inférieure à ε : les k premiers éléments de la partie décimale de $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}((u,v),(U,V))$ sont nuls. Soit k_1 le plus petit entier tel que, si $V^0 = v^0$, ..., $V^{k_1} = v^{k_1}$, alors $U^0 = u^0$, ..., $U^{\sum_{l=0}^{k_1} V^l - 1} = u^{\sum_{l=0}^{k_1} v^l - 1}$. Alors $d_{\mathbb{S}_{\mathsf{N},\mathcal{P}}}((u,v),(U,V)) < \varepsilon$. En d'autres mots, chaque y de la forme $(e,((u^0,...,u^{\sum_{l=0}^{k_1} v^l - 1}),(v^0,...,v^{k_1}))$ est dans $\mathcal{B}(x,\varepsilon)$.

Soit y^0 un tel point et $z = G^{k_1}_{f_u,\mathcal{P}}(y^0) = (e',(u',v'))$. $G_{f_u,\mathcal{P}}$ étant fortement connexe, il existe un chemin entre e' et $\check{e}$. Soit $a_0,\ldots,a_{k_2}$ les arêtes visitées le long de ce chemin. On fixe $V^{k_1} = |a_0|$ (le nombre de termes dans la séquence finie a_1), $V^{k_1+1} = |a_1|$, ..., $V^{k_1+k_2} = |a_{k_2}|$, et $U^{k_1} = a^0_0$, $U^{k_1+1} = a^1_0$, ..., $U^{k_1+V_{k_1}-1} = a_0^{V_{k_1}-1}$, $U^{k_1+V_{k_1}} = a^0_1$, $U^{k_1+V_{k_1}+1} = a^1_1$,...

Soit $y = (e,((u^0,...,u^{\sum_{l=0}^{k_1} v^l - 1}, a^0_0,...,a_0^{|a_0|}, a^0_1,...,a_1^{|a_1|},...,a^0_{k_2},...,a_{k_2}^{|a_{k_2}|}, \check{u}^0,\check{u}^1,...),(v^0,...,v^{k_1},|a_0|,...,|a_{k_2}|,\check{v}^0,\check{v}^1,...)))$. Ainsi $y \in \mathcal{B}(x,\varepsilon)$ et $G^{k_1+k_2}_f(y) = \check{x}$.

Réciproquement, si $G_{f_u,\mathcal{P}}$ n'est pas fortement connexe, il y a donc deux nœuds e_1 et e_2 sans chemins entre eux. Il n'est ainsi pas possible de trouver un couple $(u,v) \in \mathbb{S}_{\mathsf{N},\mathcal{P}}$ et $n\mathbb{N}$ tel que $G^n_{f_u,\mathcal{P}}(e,(u,v))_1 = e_2$. La boule ouverte $\mathcal{B}(e_2,1/2)$ ne peut ainsi pas être atteinte depuis n'importe quel voisins de e_1 : $G_{f_u,\mathcal{P}}$ n'est pas transitive. □

Montrons maintenant que

Lemme 9. *Si $G_{f_u,\mathcal{P}}$ est fortement connexe, alors $G_{f_u,\mathcal{P}}$ est régulière sur $(\mathcal{X}_{\mathsf{N},\mathcal{P}}, d)$.*

Démonstration. Soit $x = (e,(u,v)) \in \mathcal{X}_{\mathsf{N},\mathcal{P}}$ et $\varepsilon > 0$. Comme dans la preuve du lemme C.2, soit $k_1 \in \mathbb{N}$ tel que

$$\Big\{(e,((u^0,...,u^{v^{k_1}-1},U^0,U^1,...),(v^0,...,v^{k_1},V^0,V^1,...)) \mid$$

$$\forall i,j \in \mathbb{N}, U^i \in [\![1,\mathsf{N}]\!], V^j \in \mathcal{P}\Big\} \subset \mathcal{B}(x,\varepsilon),$$

et $y = G^{k_1}_{f_u,\mathcal{P}}(e,(u,v))$. $G_{f_u,\mathcal{P}}$ étant fortement connexe, il existe au moins un chemin entre l'état booléen y_1 de y et e. Nommons $a_0,\dots,a_{k_2}$ les arêtes d'un tel chemin. Le point

$$(e,((u^0,...,u^{v^{k_1}-1},a_0^0,...,a_0^{|a_0|},a_1^0,...,a_1^{|a_1|},...,a_{k_2}^0,...,a_{k_2}^{|a_{k_2}|},u^0,...,u^{v^{k_1}-1},$$

$$a_0^0,...,a_{k_2}^{|a_{k_2}|}...),(v^0,...,v^{k_1},|a_0|,...,|a_{k_2}|,v^0,...,v^{k_1},|a_0|,...,|a_{k_2}|,...))$$

est un point périodique dans le voisinage $\mathcal{B}(x,\varepsilon)$ de x. □

$G_{f_u,\mathcal{P}}$ étant topologiquement transitive and régulière, on peut démontrer le théorème :

Théorème [19] (Conditions pour la chaoticité de $G_{f_u,\mathcal{P}}$)**.** *La fonction $G_{f_u,\mathcal{P}}$ est chaotique sur $(\mathcal{X}_{\mathsf{N},\mathcal{P}},d)$ si et seulement si le graphe d'itérations $\mathsf{GIU}_{\mathcal{P}}(f)$ est fortement connexe.*

C.3/ CODES DE GRAY ÉQUILIBRÉS PAR INDUCTION

Théorème [22] (Existence d'un code de Gray équilibré)**.** *Soit* N *dans $\mathbb{N}^*$, et a_{N} défini par $a_{\mathsf{N}} = 2\left\lfloor \frac{2^{\mathsf{N}}}{2\mathsf{N}} \right\rfloor$. Il existe une séquence l dans l'étape (1.) de l'extension de l'algorithme de* Robinson-Cohn *telle que les nombres de transitions $TC_{\mathsf{N}}(i)$ valent tous a_{N} ou $a_{\mathsf{N}}+2$ pour chaque i, $1 \leq i \leq \mathsf{N}$.*

Démonstration. La preuve de ce théorème s'effectue par induction sur N. On peut vérifier aisément que ce théorème est établi pour les deux plus petites valeurs paires et impaires, *i.e.* pour $\mathsf{N} = 3$ et $\mathsf{N} = 4$.

Pour le cas initial où $\mathsf{N} = 3$, *i.e.* $\mathsf{N}-2 = 1$ on a : $S_1 = 1,1$, $l = 2$, $u_0 = \emptyset$ et $v = \emptyset$. Ainsi, l'algorithme produit $U = 1,2,1$, $V = 3$, $W = 2,1,1,3$ et $W' = 1,2,1,3$. Finalement, $S_3 = 1,2,1,3,1,2,1,3$ qui vérifie le théorème.

Pour le cas initial où $\mathsf{N} = 4$, *i.e.* $\mathsf{N}-2 = 2$ on a : $S_1 = 1,2,1,2$, $l = 4$, $u_0,u_1,u_2 = \emptyset,\emptyset,\emptyset$ et $v = \emptyset$. Ainsi, l'algorithme produit $U = 1,3,2,3,4,1,4,3,2$, $V = 4$, $W = 3,1,2,1,2,4$ et $W' = 1,3,2,1,2,4$. Finalement, $S_4 = 2,3,4,1,4,3,2,3,1,4,1,3,2,1,2,4$ et le code est totalement équilibré.

Pour le cas inductif, on définit tout d'abord quelques variables. Soit c_{N} (resp. d_{N}) le nombre d'éléments dont le nombre de transitions est exactement a_{N} (resp $a_{\mathsf{N}}+2$). Ces deux variables sont caractérisées par le système :

$$\begin{cases} c_{\mathsf{N}} + d_{\mathsf{N}} &= \mathsf{N} \\ c_{\mathsf{N}}a_{\mathsf{N}} + d_{\mathsf{N}}(a_{\mathsf{N}}+2) &= 2^{\mathsf{N}} \end{cases} \Leftrightarrow \begin{cases} d_{\mathsf{N}} &= \dfrac{2^{\mathsf{N}} - \mathsf{N}.a_{\mathsf{N}}}{2} \\ c_{\mathsf{N}} &= \mathsf{N} - d_{\mathsf{N}} \end{cases}$$

Puisque a_{N} est pair, d_{N} est un entier. Prouvons d'abord que c_{N} et d_{N} sont des entiers positifs. Soit q_{N} et r_{N}, définis respectivement comme étant le quotient et le reste dans la

division euclidienne de 2^N par 2N, *i.e.* $2^N = q_N.2N + r_N$, avec $0 \leq r_N < 2N$. Tout d'abord, l'entier r est pair puisque r_N est un multiple de 2 : $r_N = 2^N - q_N.2N = 2(2^{N-1} - q_N.N)$. Ensuite, a_N vaut $\frac{2^N - r_N}{N}$. Ainsi d_N vaut $r_N/2$. C'est donc un entier positif tel que $0 \leq d_N < N$. La preuve pour c_N est évidente.

Pour chaque i, $1 \leq i \leq N$, soit zi_N (resp. ti_N et bi_N) le nombre d'occurrences de l'élément i dans la séquence $u_0, \ldots, u_{l-2}$ (resp. dans les séquences $s_{i_1}, \ldots, s_{i_l}$ et v) à l'étape (1.) de l'algorithme.

En raison de la définition de u' à l'étape (2.), $3.zi_N + ti_N$ est le nombre d'occurrences de i dans la séquence U. Il est évident que le nombre d'occurrences de i dans la séquence V est $2bi_N$ en raison de l'étape (3.). On a ainsi le système suivant :

$$\left\{ \begin{array}{lcl} 3.zi_N + ti_N + 2.bi_N + TC_{N-2}(i) & = & TC_N(i) \\ zi_N + ti_N + bi_N & = & TC_{N-2}(i) \end{array} \right. \Leftrightarrow$$

$$\left\{ \begin{array}{lcl} zi_N & = & \dfrac{TC_N(i) - 2.TC_{N-2}(i) - bi_N}{2} \\ ti_N & = & TC_{N-2}(i) - zi_N - bi_N \end{array} \right. \quad \text{(C.1)}$$

Dans ce système de 2 équations à trois inconnues, on pose $b_i = 0$. Puisque TC_N est pair (égal à a_N ou à $a_N + 2$), l'inconnue zi_N est donc un entier. Prouvons alors que le système résultant admet toujours une solution positive pour z_i et pour t_i telle $0 \leq z_i, t_i \leq TC_{N-2}(i)$ et telle que leur somme est égale à $TC_{N-2}(i)$. On remarque que cette contrainte est toujours établie si le système admet une solution. On a donc le système suivant :

$$\left\{ \begin{array}{lcl} zi_N & = & \dfrac{TC_N(i) - 2.TC_{N-2}(i)}{2} \\ ti_N & = & TC_{N-2}(i) - zi_N \end{array} \right. \quad \text{(C.2)}$$

La définition de $TC_N(i)$ dépend de la valeur de N. Pour $3 \leq N \leq 7$, on définit celles-ci comme suit :

$$\begin{array}{lcl} TC_3 & = & [2, 2, 4] \\ TC_5 & = & [6, 6, 8, 6, 6] \\ TC_7 & = & [18, 18, 20, 18, 18, 18, 18] \end{array}$$

$$\begin{array}{lcl} TC_4 & = & [4, 4, 4, 4] \\ TC_6 & = & [10, 10, 10, 10, 12, 12] \end{array}$$

Il n'est pas difficile de vérifier que dans chacun des cas précédents, le système admet bien une solution.

Pour $N \geq 8$, on définit $TC_N(i)$ comme suit :

$$TC_N(i) = \left\{ \begin{array}{l} a_N \text{ si } 1 \leq i \leq c_N \\ a_N + 2 \text{ si } c_N + 1 \leq i \leq c_N + d_N \end{array} \right. \quad \text{(C.3)}$$

On a ainsi

$$\begin{aligned} TC_{\mathsf{N}}(i) - 2.TC_{\mathsf{N}-2}(i) &\geq a_{\mathsf{N}} - 2(a_{\mathsf{N}-2}+2) \\ &\geq \frac{2^{\mathsf{N}}-r_{\mathsf{N}}}{\mathsf{N}} - 2\left(\frac{2^{\mathsf{N}-2}-r_{\mathsf{N}-2}}{\mathsf{N}-2}+2\right) \\ &\geq \frac{2^{\mathsf{N}}-2N}{\mathsf{N}} - 2\left(\frac{2^{\mathsf{N}-2}}{\mathsf{N}-2}+2\right) \\ &\geq \frac{(\mathsf{N}-2).2^{\mathsf{N}}-2N.2^{\mathsf{N}-2}-6N(N-2)}{\mathsf{N}.(\mathsf{N}-2)} \end{aligned}$$

Une simple étude de variation de la fonction $t : \mathbb{R} \to \mathbb{R}$ telle que $x \mapsto t(x) = (x-2).2^x - 2x.2^{x-2} - 6x(x-2)$ montre que sa dérivée est strictement positive pour $x \geq 6$ et que $t(8) = 224$. L'entier $TC_{\mathsf{N}}(i) - 2.TC_{\mathsf{N}-2}(i)$ est ainsi positif pour chaque $\mathsf{N} \geq 8$, ce qui termine la preuve. □

C.4/ MAJORATION DU TEMPS DE MIXAGE

L'objectif principal de cette section est de démontrer le théorème 23 rappelé en fin de section.

Un résultat classique est

$$t_{\text{mix}}(\varepsilon) \leq \lceil \log_2(\varepsilon^{-1}) \rceil t_{\text{mix}}(\frac{1}{4})$$

Soit $(X_t)_{t\in\mathbb{N}}$ une suite de variables aléatoires de $\mathbb{B}^{\mathsf{N}}$. Une variable aléatoire τ dans $\mathbb{N}$ est un *temps d'arrêt* pour la suite (X_i) si pour chaque t il existe $B_t \subseteq (\mathbb{B}^{\mathsf{N}})^{t+1}$ tel que $\{\tau = t\} = \{(X_0, X_1, \ldots, X_t) \in B_t\}$. En d'autres termes, l'événement $\{\tau = t\}$ dépend uniquement des valeurs de $(X_0, X_1, \ldots, X_t)$, et non de celles de X_k pour $k > t$.

Soit $(X_t)_{t\in\mathbb{N}}$ une chaîne de Markov et $f(X_{t-1}, Z_t)$ une représentation fonctionnelle de celle-ci. Un *temps d'arrêt aléatoire* pour la chaîne de Markov est un temps d'arrêt pour $(Z_t)_{t\in\mathbb{N}}$. Si la chaîne de Markov est irréductible et a π comme distribution stationnaire, alors un *temps stationnaire* τ est temps d'arrêt aléatoire (qui peut dépendre de la configuration initiale X), tel que la distribution de X_τ est π :

$$\mathbb{P}_X(X_\tau = Y) = \pi(Y).$$

Un temps d'arrêt τ est qualifié de *fort* si X_τ est indépendant de τ.

On rappelle le théorème suivant [LPW06, Proposition 6.10].

Théorème [28]. *Si τ est un temps d'arrêt fort, alors $d(t) \leq \max_{X\in\mathbb{B}^{\mathsf{N}}} \mathbb{P}_X(\tau > t)$.*

Soit $E = \{(X, Y) \mid X \in \mathbb{B}^{\mathsf{N}}, Y \in \mathbb{B}^{\mathsf{N}},\ X = Y \text{ ou } X \oplus Y \in 0^*10^*\}$. En d'autres mots, E est l'ensemble des tous les arcs du N-cube. Soit $h : \mathbb{B}^{\mathsf{N}} \to [\mathsf{N}]$ qui mémorise pour chaque nœud $X \in \mathbb{B}^{\mathsf{N}}$ quel arc est supprimé à partir du cycle hamiltonien, *i.e.* quel bit dans $[\mathsf{N}]$ ne peut pas être inversé.

On définit ensuite l'ensemble $E_h = E \setminus \{(X, Y) \mid X \oplus Y = 0^{\mathsf{N}-h(X)}10^{h(X)-1}\}$. C'est l'ensemble de tous les arcs appartenant à l'hypercube modifié, *i.e.*, le N-cube où le cycle hamiltonien h a été enlevé.

On définit la matrice de Markov P_h pour chaque ligne X et chaque colonne Y comme suit :

$$\begin{cases} P_h(X,X) = \frac{1}{2} + \frac{1}{2\mathsf{N}} & \\ P_h(X,Y) = 0 & \text{si } (X,Y) \notin E_h \\ P_h(X,Y) = \frac{1}{2\mathsf{N}} & \text{si } X \neq Y \text{ et } (X,Y) \in E_h \end{cases} \tag{C.4}$$

Soit alors $\overline{h} : \mathbb{B}^{\mathsf{N}} \to \mathbb{B}^{\mathsf{N}}$ la fonction telle que pour $X \in \mathbb{B}^{\mathsf{N}}$, $(X, \overline{h}(X)) \in E$ et $X \oplus \overline{h}(X) = 0^{\mathsf{N}-h(X)}10^{h(X)-1}$. La fonction $\overline{h}$ est dite *anti-involutive* si pour tout $X \in \mathbb{B}^{\mathsf{N}}$, $\overline{h}(\overline{h}(X)) \neq X$.

Lemme 10. *Si $\overline{h}$ est bijective et anti-involutive, alors $h(\overline{h}^{-1}(X)) \neq h(X)$.*

Démonstration. Soit $\overline{h}$ bijective. Soit $k \in [\![1, \mathsf{N}]\!]$ t.q. $h(\overline{h}^{-1}(X)) = k$. Alors $(\overline{h}^{-1}(X), X)$ appartient à E et $\overline{h}^{-1}(X) \oplus X = 0^{\mathsf{N}-k}10^{k-1}$. Supposons $h(X) = h(\overline{h}^{-1}(X))$. Dans un tel cas, $h(X) = k$. Par définition $\overline{h}$, $(X, \overline{h}(X)) \in E$ et $X \oplus \overline{h}(X) = 0^{\mathsf{N}-h(X)}10^{h(X)-1} = 0^{\mathsf{N}-k}10^{k-1}$. Ainsi $\overline{h}(X) = \overline{h}^{-1}(X)$, ce qui entraîne $\overline{h}(\overline{h}(X)) = X$ et qui contredit le fait que $\overline{h}$ est anti-involutive. □

Soit Z une variable aléatoire suivant une distribution uniforme sur $[\mathsf{N}] \times \mathbb{B}$. Pour $X \in \mathbb{B}^{\mathsf{N}}$, on définit avec $Z = (i, b)$,

$$\begin{cases} f(X,Z) = X \oplus (0^{\mathsf{N}-i}10^{i-1}) & \text{si } b = 1 \text{ et } i \neq h(X), \\ f(X,Z) = X & \text{sinon.} \end{cases}$$

La chaîne de Markov est ainsi définie par

$$X_t = f(X_{t-1}, Z_t).$$

Un entier $\ell \in [\![1, \mathsf{N}]\!]$ est *équitable* au temps t s'il existe $0 \leq j < t$ tel que $Z_{j+1} = (\ell, \cdot)$ et $h(X_j) \neq \ell$. En d'autres mots, il existe une date j antérieure à t où le premier élément de la variable aléatoire Z est l (*i.e.*, la stratégie vaut l à la date j) et où le nœud X_j permet de traverser l'arc l.

Soit τ_{stop} le premier temps où tous les éléments de $[\![1, \mathsf{N}]\!]$ sont équitables. On a le lemme suivant :

Lemme 11. *L'entier τ_{stop} est un temps stationnaire fort.*

Démonstration. Soit τ_ℓ la première date où ℓ est équitable. La variable aléatoire Z_{τ_ℓ} est de la forme (ℓ, b) et telle que $b = 1$ avec la probabilité $\frac{1}{2}$ et $b = 0$ avec la probabilité $\frac{1}{2}$. Comme $h(X_{\tau_\ell - 1}) \neq \ell$, la valeur du $\ell^{\text{ème}}$ bit de X_{τ_ℓ} est 0 ou 1 avec la même probabilité ($\frac{1}{2}$).

A chaque étape, le $l^{\text{ème}}$ bit est inversé de 0 à 1 ou de 1 à 0, chaque fois avec la même probabilité. Ainsi, pour $t \geq \tau_\ell$, le $\ell^{\text{ème}}$ bit de X_t est 0 ou 1 avec la même probabilité, et ce indépendamment de la valeur des autres bits. □

Pour chaque $X \in \mathbb{B}^{\mathsf{N}}$ et $\ell \in [\mathsf{N}]$, soit $S_{X,\ell}$ une variable aléatoire qui compte le nombre d'itérations tel qu'en démarrant de la configuration X on en atteigne une où ℓ est équitable. Plus formellement,

$$S_{X,\ell} = \min\{t \geq 1 \mid h(X_{t-1}) \neq \ell \text{ et } Z_t = (\ell, .) \text{ et } X_0 = X\}.$$

On peut majorer l'espérance de cette variable aléatoire comme suit.

Lemme 12. *Soit $\overline{h}$ un fonction bijective et anti-involutive. Alors pour tout $X \in \mathbb{B}^{\mathsf{N}}$ et $\ell \in [\mathsf{N}]$, l'inégalité $E[S_{X,\ell}] \leq 8\mathsf{N}^2$ est établie.*

Démonstration. Montrons tout d'abord que pour chaque X et chaque ℓ, on a $\mathbb{P}(S_{X,\ell}) \leq 2) \geq \frac{1}{4\mathsf{N}^2}$. Soit $X_0 = X$.

- Si $h(X) \neq \ell$, alors $\mathbb{P}(S_{X,\ell} = 1) = \frac{1}{2\mathsf{N}} \geq \frac{1}{4\mathsf{N}^2}$.
- Sinon, $h(X) = \ell$, alors $\mathbb{P}(S_{X,\ell} = 1) = 0$. Dans ce cas, intuitivement, il est possible de passer de X à $\overline{h}^{-1}(X)$ (avec la probabilité $\frac{1}{2N}$). De plus, dans la configuration $\overline{h}^{-1}(X)$, le $l^{\text{ème}}$ bit peut être inversé. Plus formellement, puisque $\overline{h}$ est anti-involutive, $\overline{h}(X) = \overline{h}(\overline{h}(\overline{h}^{-1}(X))) \neq \overline{h}^{-1}(X)$. On en déduit que $(X, \overline{h}^{-1}(X)) \in E_h$. On a ainsi $P(X_1 = \overline{h}^{-1}(X)) = \frac{1}{2\mathsf{N}}$. D'après le lemme 10, $h(\overline{h}^{-1}(X))$ est différent de $h(X)$. Ainsi, $\mathbb{P}(S_{x,\ell} = 2 \mid X_1 = \overline{h}^{-1}(X)) = \frac{1}{2\mathsf{N}}$, prouvant ainsi que $\mathbb{P}(S_{x,\ell} \leq 2) \geq \frac{1}{4\mathsf{N}^2}$.

Ainsi, $\mathbb{P}(S_{X,\ell} \geq 3) \leq 1 - \frac{1}{4\mathsf{N}^2}$. Par induction, on a, pour chaque i, $\mathbb{P}(S_{X,\ell} \geq 2i) \leq \left(1 - \frac{1}{4\mathsf{N}^2}\right)^i$. De plus, puisque $S_{X,\ell}$ est positive, on sait [MU05, lemme 2.9] que

$$E[S_{X,\ell}] = \sum_{i=1}^{+\infty} \mathbb{P}(S_{X,\ell} \geq i).$$

Puisque $\mathbb{P}(S_{X,\ell} \geq i) \geq \mathbb{P}(S_{X,\ell} \geq i+1)$, on a

$$E[S_{X,\ell}] = \sum_{i=1}^{+\infty} \mathbb{P}(S_{X,\ell} \geq i) \leq \mathbb{P}(S_{X,\ell} \geq 1) + \mathbb{P}(S_{X,\ell} \geq 2) + 2\sum_{i=1}^{+\infty} \mathbb{P}(S_{X,\ell} \geq 2i).$$

Ainsi,

$$E[S_{X,\ell}] \leq 1 + 1 + 2\sum_{i=1}^{+\infty} \left(1 - \frac{1}{4\mathsf{N}^2}\right)^i = 2 + 2(4\mathsf{N}^2 - 1) = 8\mathsf{N}^2,$$

ce qui conclut la preuve du lemme. □

Soit τ'_{stop} la variable aléatoire comptant le nombre d'itérations pour avoir exactement $\mathsf{N} - 1$ bits équitables. On peut majorer son espérance.

Lemme 13. *On a $E[\tau'_{\text{stop}}] \leq 4\mathsf{N}\ln(\mathsf{N}+1)$.*

Démonstration. C'est un problème classique de collectionneur de vignettes. Soit W_i la variable aléatoire comptant le nombre de déplacements dans la chaîne de Markov pour avoir exactement $i-1$ bits équitables. On a $\tau'_{\text{stop}} = \sum_{i=1}^{\mathsf{N}-1} W_i$. Dans la configuration X avec $i-1$ bits équitables, la probabilité d'obtenir un nouveau bit équitable est soit $1 - \frac{i-1}{\mathsf{N}}$ si $h(X)$ est équitable, soit $1 - \frac{i-2}{\mathsf{N}}$ si $h(X)$ ne l'est pas.

Ainsi, $\mathbb{P}(W_i = k) \leq \left(\frac{i-1}{\mathsf{N}}\right)^{k-1} \frac{\mathsf{N}-i+2}{\mathsf{N}}$. On a par conséquent $\mathbb{P}(W_i \geq k) \leq \left(\frac{i-1}{\mathsf{N}}\right)^{k-1} \frac{\mathsf{N}-i+2}{\mathsf{N}-i+1}$. On en déduit que

$$E[W_i] = \sum_{k=1}^{+\infty} \mathbb{P}(W_i \geq k) \leq \mathsf{N}\frac{\mathsf{N}-i+2}{(\mathsf{N}-i+1)^2} \leq \frac{4\mathsf{N}}{\mathsf{N}-i+2},$$

et donc que

$$E[\tau'_{\text{stop}}] = \sum_{i=1}^{\mathsf{N}-1} E[W_i] \leq 4\mathsf{N} \sum_{i=1}^{\mathsf{N}-1} \frac{1}{\mathsf{N}-i+2} = 4\mathsf{N} \sum_{i=3}^{\mathsf{N}+1} \frac{1}{i}.$$

Or $\sum_{i=1}^{\mathsf{N}+1} \frac{1}{i} \leq 1 + \ln(\mathsf{N}+1)$. On en déduit que $1 + \frac{1}{2} + \sum_{i=3}^{\mathsf{N}+1} \frac{1}{i} \leq 1 + \ln(\mathsf{N}+1)$. Ainsi, on a la majoration suivante

$$E[\tau'_{\text{stop}}] \leq 4\mathsf{N}(-\frac{1}{2} + \ln(\mathsf{N}+1)) \leq 4\mathsf{N}\ln(\mathsf{N}+1),$$

ce qui termine la preuve du lemme. □

Tous les éléments sont en place pour majorer l'espérance de τ_{stop}.

Théorème 29. *Si $\overline{h}$ est bijective et anti- involutive $\overline{h}(\overline{h}(X)) \neq X$, alors $E[\tau_{\text{stop}}] \leq 8\mathsf{N}^2 + 4\mathsf{N}\ln(\mathsf{N}+1)$.*

Démonstration. Sans perte de généralité, considérons que le dernier bit non équitable est ℓ. On a $\tau_{\text{stop}} = \tau'_{\text{stop}} + S_{X_\tau,\ell}$ et donc $E[\tau_{\text{stop}}] = E[\tau'_{\text{stop}}] + E[S_{X_\tau,\ell}]$. Ainsi, le théorème est une application directe des lemmes 12 et 13. □

Montrons alors le théorème 23 rappelé ci-dessous

Théorème 23 (Temps de mixage sans chemin hamiltonien)**.** *On considère un* N*-cube dans lequel un chemin hamiltonien a été supprimé et la fonction de probabilités p définie sur l'ensemble des arcs comme suit :*

$$p(e)\begin{cases} = \frac{1}{2} + \frac{1}{2\mathsf{N}} \textit{ si } e = (v,v) \textit{ avec } v \in \mathbb{B}^{\mathsf{N}}, \\ = \frac{1}{2\mathsf{N}} \textit{ sinon.} \end{cases}$$

La chaîne de Markov associée converge vers la distribution uniforme et

$$\forall \varepsilon > 0,\ t_{\text{mix}}(\varepsilon) \leq x \leq \lceil \log_2(\varepsilon^{-1})(32\mathsf{N}^2 + 16\mathsf{N}\ln(\mathsf{N}+1))$$

Démonstration. Comme τ est positive, on peut appliquer l'inégalité de Markov :

$$\mathbb{P}_X(\tau > t) \leq \frac{E[\tau]}{t}.$$

En posant $t_n = 32N^2 + 16N\ln(N+1)$, on obtient :

$$\mathbb{P}_X(\tau > t_n) \leq \frac{1}{4}.$$

D'après la définition de t_{mix} et d'après le théorème 28, on en déduit

$$t_{\text{mix}} \leq 32N^2 + 16N\ln(N+1) = O(N^2)$$

. □

D

PREUVES SUR LE MARQUAGE DE MÉDIA

D.1/ LE MARQUAGE EST ϵ-STÉGO-SÉCURE

Prouvons le théorème suivant.

Théorème [25] (ε-stego sécurité)**.** *Soit ε un nombre positif, l un nombre de LSBs, $X \sim \mathbf{U}\left(\mathbb{B}^l\right)$, un adapteur de stratégie uniformément distribué indépendant de X f_l un mode tel que* GIU(f_l) *est fortement connexe et la matrice de Markov associée à f_l est doublement stochastique. Il existe un nombre q d'itérations tel que $|p(Y|K) - p(X)| < \varepsilon$.*

Démonstration. Soit *deci* la bijection entre $\mathbb{B}^l$ et $[\![0, 2^l - 1]\!]$ qui associe la valeur décimale à chaque nombre binaire dans $\mathbb{B}^l$. La probabilité $p(X^t) = (p(X^t = e_0), \ldots, p(X^t = e_{2^l-1}))$ pour $e_j \in \mathbb{B}^l$ est égale à $(p(\textit{deci}(X^t) = 0, \ldots, p(\textit{deci}(X^t) = 2^l - 1))$, notée par la suite π^t. Pour $i \in [\![0, 2^l - 1]\!]$, la probabilité $p(\textit{deci}(X^{t+1}) = i)$ est

$$\sum_{j=0}^{2^l-1} \sum_{k=1}^{l} p(\textit{deci}(X^t) = j, S^t = k, i =_k j, f_k(j) = i_k)$$

où $i =_k j$ est vraie si et seulement si les représentations binaires de i et de j ne diffèrent que pour le $k^{\text{ème}}$ élément et où i_k représente dans cette preuve le $k^{\text{ème}}$ élément dans la représentation binaire du nombre i.

En raison des hypothèses sur la stratégie, la probabilité $p(\textit{deci}(X^t) = j, S^t = k, i =_k j, f_k(j) = i_k)$ est égale à $\frac{1}{l}.p(\textit{deci}(X^t) = j, i =_k j, f_k(j) = i_k)$. Enfin, puisque $i =_k j$ et $f_k(j) = i_k$ sont constants et sont donc indépendants de X^t, on a

$$\pi_i^{t+1} = \sum_{j=0}^{2^l-1} \pi_j^t . \frac{1}{l} \sum_{k=1}^{l} p(i =_k j, f_k(j) = i_k).$$

Puisque $\frac{1}{l} \sum_{k=1}^{l} p(i =_k j, f_k(j) = i_k)$ est égal à M_{ji} où M est la matrice de Markov associée à f_l, on a ainsi

$$\pi_i^{t+1} = \sum_{j=0}^{2^l-1} \pi_j^t . M_{ji} \text{ et donc } \pi^{t+1} = \pi^t M.$$

Maintenant, puisque le graphe $\Gamma(f)$ est fortement connexe, pour chaque couple de sommets (i, j), un chemin peut être trouvé de i jusqu'à j de longueur au plus égale à 2^l. Il existe donc $k_{ij} \in [\![1, 2^l]\!]$ t.q. $M_{ij}^{k_{ij}} > 0$.

Comme tous les multiples $l \times k_{ij}$ de k_{ij} sont tels que $M_{ij}^{l \times k_{ij}} > 0$, on peut conclure que si k est le PPCM de $\left\{k_{ij}/i, j \in [\![1, 2^l]\!]\right\}$ alors $\forall i, j \in [\![1, 2^l]\!], M_{ij}^k > 0$ et donc M est une matrice stochastique régulière.

Théorème [30]. *Si M est une matrice stochastique régulière, alors M possède un unique vecteur stationnaire de probabilités π ($\pi.M = \pi$). De plus, si π^0 est un vecteur de probabilité et si on définit la suite $(\pi^k)^{k\in\mathbb{N}}$ par $\pi^{k+1} = \pi^k.M$ pour $k = 0, 1, \ldots$ alors la chaîne de Markov π^k converge vers π lorsque k tend vers l'infini.*

Grâce à ce théorème, M admet un unique vecteur stationnaire de probabilité π. Par hypothèses, puisque M est doublement stochastique, on a $(\frac{1}{2^l}, \ldots, \frac{1}{2^l}) = (\frac{1}{2^l}, \ldots, \frac{1}{2^l})M$ et donc $\pi = (\frac{1}{2^l}, \ldots, \frac{1}{2^l})$. Il existe donc q t.q. $|\pi^q - \pi| < \epsilon$. Puisque $p(Y|K)$ est $p(X^q)$, la méthode est donc ϵ-stégo-sécure pour peu que l'adapteur de stratégie soit uniformément distribué. □

D.2/ LE MODE f_l EST DOUBLEMENT STOCHASTIQUE

On considère le mode $f_l : \mathbb{B}^l \to \mathbb{B}^l$ t.q. le $i^{\text{ème}}$ composant est défini par

$$f_l(x)_i = \begin{cases} \overline{x_i} \text{ si } i \text{ est impair} \\ x_i \oplus x_{i-1} \text{ si } i \text{ est pair} \end{cases} \qquad (^{(7.3)})$$

Prouvons que la matrice de Markov associée est doublement stochastique par induction sur la longueur l. Pour $l = 1$ et $l = 2$ la preuve est évidente. Considérons que le résultat est établi jusqu'à $l = 2k$ avec $k \in \mathbb{N}$.

On montre d'abord que la double stochasticité est établie pour $l = 2k + 1$. En suivant les notations introduites à la section B.3, soit $\text{GIU}(f_{2k+1})^0$ et $\text{GIU}(f_{2k+1})^1$ les sous-graphes de $\text{GIU}(f_{2k+1})$ induits par les ensembles $\mathbb{B}^{2k} \times \{0\}$ et $\mathbb{B}^{2k} \times \{1\}$ de $\mathbb{B}^{2k+1}$ respectivement. Les graphes $\text{GIU}(f_{2k+1})^0$ et $\text{GIU}(f_{2k+1})^1$ sont isomorphes à $\text{GIU}(f_{2k})$. De plus, ils ne sont liés dans $\text{GIU}(f_{2k+1})$ que par des arcs de la forme $(x_1, \ldots, x_{2k}, 0) \to (x_1, \ldots, x_{2k}, 1)$ et $(x_1, \ldots, x_{2k}, 1) \to (x_1, \ldots, x_{2k}, 0)$. Dans $\text{GIU}(f_{2k+1})$, deux sortes d'arcs pointent vers $(x_1, \ldots, x_{2k}, 0)$. Ceux qui sont de la forme $(y_1, \ldots, y_{2k}, 0)$, où un seul des y_i est différent de x_i, et leur nombre est celui des arcs qui pointent vers $(x_1, \ldots, x_{2k})$ dans $\text{GIU}(f_{2k})$. L'arc $(x_1, \ldots, x_{2k}, 0) \to (x_1, \ldots, x_{2k}, 0)$ qui existe d'après la définition de f_l. De même pour le nombre d'arcs dont l'extrémité est de la forme $(x_1, \ldots, x_{2k}, 1)$. Par hypothèse d'induction, la chaîne de Markov associée à $\text{GIU}(f_{2k})$ est doublement stochastique. Ainsi tous les sommets $(x_1, \ldots, x_{2k})$ ont le même nombre d'arcs entrants et la preuve est établie pour $l = 2k + 1$.

Montrons à présent la double stochasticité pour $l = 2k + 2$. La fonction f_l est définie par $f_l(x) = (\overline{x_1}, x_2 \oplus x_1, \ldots, \overline{x_{2k+1}}, x_{2k+2} \oplus x_{2k+1})$. On se concentre sur $\text{GIU}(f_{2k+2})^0$ et $\text{GIU}(f_{2k+2})^1$ qui sont isomorphes à $\text{GIU}(f_{2k+1})$. Parmi les configurations de $\mathbb{B}^{2k+2}$, seuls quatre suffixes de longueur 2 peuvent apparaître : 00, 10, 11 et 01. Puisque $f_{2k+2}(\ldots, 0, 0)_{2k+2} = 0$, $f_{2k+2}(\ldots, 1, 0)_{2k+2} = 1$, $f_{2k+2}(\ldots, 1, 1)_{2k+2} = 0$ et $f_{2k+2}(\ldots, 0, 1)_{2k+2} = 1$, le nombre d'arcs dont les extrémités sont

— $(x_1, \ldots, x_{2k}, 0, 0)$ est le même que celui dont l'extrémité est de la forme $(x_1, \ldots, x_{2k}, 0)$ dans GIU(f_{2k+1}) auquel on ajoute 1 (une boucle autour des configurations $(x_1, \ldots, x_{2k}, 0, 0)$) ;
— $(x_1, \ldots, x_{2k}, 1, 0)$ est le même que celui dont l'extrémité est de la forme $(x_1, \ldots, x_{2k}, 0)$ in GIU(f_{2k+1}) auquel on ajoute 1 (l'arc entre les configurations $(x_1, \ldots, x_{2k}, 1, 1)$ et les configurations $(x_1, \ldots, x_{2k}, 1, 0)$) ;
— $(x_1, \ldots, x_{2k}, 0, 1)$ est le même que celui dont l'extrémité est de la forme $(x_1, \ldots, x_{2k}, 0)$ in GIU(f_{2k+1}) auquel on ajoute 1 (une boucle autour des configurations $(x_1, \ldots, x_{2k}, 0, 1)$) ;
— $(x_1, \ldots, x_{2k}, 1, 1)$ est le même que celui dont l'extrémité est de la forme $(x_1, \ldots, x_{2k}, 1)$ in GIU(f_{2k+1}) auquel on ajoute 1 (l'arc entre les configurations $(x_1, \ldots, x_{2k}, 1, 0)$ et les configurations $(x_1, \ldots, x_{2k}, 1, 1)$).

Chacun des sommets $(x_1, \ldots, x_{2k+2})$ a donc le même nombre d'arcs entrants, la preuve est donc établie pour $l = 2k + 2$.

D.3/ LE MARQUAGE EST CORRECT ET COMPLET

Théorème [26] (Correction et complétude du marquage)**.** *La condition de l'algorithme de marquage est nécessaire et suffisante pour permettre l'extraction du message du média marqué.*

Démonstration. Pour la suffisance, soit d_i la dernière itération où l'élément $i \in \mathfrak{I}(S_p)$ de la configuration x a été modifié :

$$d_i = \max\{j | S_p^j = i\}.$$

Soit $D = \{d_i | i \in \mathfrak{I}(S_p)\}$. L'ensemble $\mathfrak{I}(S_c)_{|D}$ est donc la restriction de l'image de S_c à D.

Le vecteur qui résulte de ces itérations est donc $(x_0^l, \ldots, x_{\mathsf{N}-1}^l)$ où x_i^l est soit $x_i^{d_i}$ si i appartient à $\mathfrak{I}(S_p)$ ou x_i^0 sinon. De plus, pour chaque $i \in \mathfrak{I}(S_p)$, l'élément $x_i^{d_i}$ est égal à $m_{S_c^{d_i}}^{d_i-1}$. Sous hypothèse que la contrainte imposée soit réalisée, tous les indices $j \in [\![0; \mathsf{P}-1]\!]$ appartiennent à $\mathfrak{I}(S_c)_{|D}$. On a alors $j \in [\![0; \mathsf{P}-1]\!]$ tel que $S_c^{d_i} = j$. On retrouve ainsi tous les éléments m_j du vecteur m. A partir de $m_j^{d_i-1}$, la valeur de m_j^0 peut être déduite en comptant dans S_c combien de fois l'élément j a été invoqué avant $d_i - 1$.

Réciproquement, si $\mathfrak{I}(S_c)_{|D} \subsetneq [\![0; \mathsf{P}-1]\!]$, il existe un $j \in [\![0; \mathsf{P}-1]\!]$ qui n'appartient pas à $\mathfrak{I}(S_c)_{|\mathfrak{I}(S_p)}$. Ainsi, m_j n'est pas présent dans x^l et le message ne peut pas extrait. □

BIBLIOGRAPHIE

[AAG+15] Bassam AlKindy, Bashar Al-Nuaimi, Christophe Guyeux, Jean-François Couchot, Michel Salomon, Reem Alsrraj, and Laurent Philippe. Binary particle swarm optimization versus hybrid genetic algorithm for inferring well supported phylogenetic trees. In Claudia Angelini, Paola M. V. Rancoita, and Stefano Rovetta, editors, *Computational Intelligence Methods for Bioinformatics and Biostatistics - 12th International Meeting, CIBB 2015, Naples, Italy, September 10-12, 2015, Revised Selected Papers*, volume 9874 of *Lecture Notes in Computer Science*, pages 165–179. Springer, 2015.

[ABCVS05] A. Abbas, J. M. Bahi, S. Contassot-Vivier, and M. Salomon. Mixing synchronism / asynchronism in discrete-state discrete-time dynamic networks. In *4th Int. Conf. on Engineering Applications and Computational Algorithms, DCDIS'2005*, pages 524–529, Guelph, Canada, July 2005. ISSN 1492-8760.

[ACGS13] Bassam Alkindy, Jean-François Couchot, Christophe Guyeux, and Michel Salomon. Finding the core-genes of chloroplast species. Journées SeqBio 2013, Montpellier, November 2013.

[And73] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[Bah00] J. M. Bahi. Boolean totally asynchronous iterations. *International Journal of Mathematical Algorithms*, 1 :331–346, 2000.

[BBCS92] John Banks, J. Brooks, G. Cairns, and P. Stacey. On devaney's definition of chaos. *Amer. Math. Monthly*, 99 :332–334, 1992.

[BBS83] Lenore Blum, Manuel Blum, and Mike Shub. Comparison of two pseudo-random number generators. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology : Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, pages 61–78, New York, NY, USA, August 1983. Plenum Press.

[BCC+15] B. Al Bouna, J. F. Couchot, R. Couturier, Y. A. Fadil, and C. Guyeux. Performance study of steganalysis techniques. In *Applied Research in Computer Science and Engineering (ICAR), 2015 International Conference on*, pages 1–7, Lebanon, October 2015.

[BCF+13] Jacques Bahi, Jean-François Couchot, Nicolas Friot, Christophe Guyeux, and Kamel Mazouzi. Quality studies of an invisible chaos-based watermarking scheme with message extraction. In *IIHMSP'13, 9th Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, pages 547–550, Beijing, China, October 2013.

[BCFG12a] Jacques Bahi, Jean-François Couchot, Nicolas Friot, and Christophe Guyeux. Application of steganography for anonymity through the internet. In *IHTIAP'2012, 1-st Workshop on Information Hiding Techniques for Internet Anonymity and Privacy*, pages 96–101, Venice, Italy, June 2012.

[BCFG12b] Jacques Bahi, Jean-François Couchot, Nicolas Friot, and Christophe Guyeux. A robust data hiding process contributing to the development of a semantic web. In *INTERNET'2012, 4-th Int. Conf. on Evolving Internet*, pages 71–76, Venice, Italy, June 2012.

[BCG11a] Jacques Bahi, Jean-François Couchot, and Christophe Guyeux. Performance analysis of a keyed hash function based on discrete and chaotic proven iterations. In *INTERNET 2011, the 3-rd Int. Conf. on Evolving Internet*, pages 52–57, Luxembourg, Luxembourg, June 2011. Best paper award.

[BCG11b] Jacques Bahi, Jean-François Couchot, and Christophe Guyeux. Steganography : a class of algorithms having secure properties. In *IIH-MSP-2011, 7-th Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, pages 109–112, Dalian, China, October 2011.

[BCG12a] Jacques Bahi, Jean-François Couchot, and Christophe Guyeux. Quality analysis of a chaotic proven keyed hash function. *International Journal On Advances in Internet Technology*, 5(1) :26–33, 2012.

[BCG12b] Jacques Bahi, Jean-François Couchot, and Christophe Guyeux. Steganography : a class of secure and robust algorithms. *The Computer Journal*, 55(6) :653–666, 2012.

[BCG16] Mohammed Bakiri, Jean-François Couchot, and Christophe Guyeux. FPGA implementation of f2-linear pseudorandom number generators based on zynq mpsoc : A chaotic iterations post processing case study. In Christian Callegari, Marten van Sinderen, Panagiotis G. Sarigiannidis, Pierangela Samarati, Enrique Cabello, Pascal Lorenz, and Mohammad S. Obaidat, editors, *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016) - Volume 4 : SECRYPT, Lisbon, Portugal, July 26-28, 2016.*, pages 302–309. SciTePress, 2016.

[BCGR11] Jacques Bahi, Jean-François Couchot, Christophe Guyeux, and Adrien Richard. On the link between strongly connected iteration graphs and chaotic boolean discrete-time dynamical systems. In *FCT'11, 18th Int. Symp. on Fundamentals of Computation Theory*, volume 6914 of *LNCS*, pages 126–137, Oslo, Norway, August 2011.

[BCGS12] Jacques Bahi, Jean-François Couchot, Christophe Guyeux, and Michel Salomon. Neural networks and chaos : Construction, evaluation of chaotic networks, and prediction of chaos with multilayer feedforward network. *Chaos, An Interdisciplinary Journal of Nonlinear Science*, 22(1) :013122–1 – 013122–9, March 2012. 9 pages.

[BCGW11] Jacques Bahi, Jean-François Couchot, Christophe Guyeux, and Qianxue Wang. Class of trustworthy pseudo random number generators. In *INTERNET 2011, the 3-rd Int. Conf. on Evolving Internet*, pages 72–77, Luxembourg, Luxembourg, June 2011.

[BCV02] J. M. Bahi and S. Contassot-Vivier. Stability of fully asynchronous discrete-time discrete-state dynamic networks. *IEEE Transactions on Neural Networks*, 13(6) :1353–1363, 2002.

[BCVC10] J. M. Bahi, S. Contassot-Vivier, and J.-F. Couchot. Convergence results of combining synchronism and asynchronism for discrete-state discrete-time dynamic network. Research Report RR2010-02, LIFC - Laboratoire d'Informatique de l'Université de Franche Comté, May 2010.

[BDCC15] Ahmad W. Bitar, Rony Darazi, Jean-François Couchot, and Raphaël Couturier. Blind digital watermarking in pdf documents using spread transform dither modulation. *Multimedia Tools and Applications*, pages 1–19, 2015.

[BG85] Manuel Blum and Shafi Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 289–302, Berlin, August 1985. Springer.

[BHJM07] Dirk Beyer, Thomas A. Henzinger, Ranjit Jhala, and Rupak Majumdar. The software model checker blast. *International Journal on Software Tools for Technology Transfer*, 9(5-6) :505–525, 2007.

[BM99] J. M. Bahi and C. Michel. Simulations of asynchronous evolution of discrete systems. *Simulation Practice and Theory*, 7 :309–324, 1999.

[BS96] Girish S. Bhat and Carla D. Savage. Balanced gray codes. *Electr. J. Comb.*, 3(1), 1996.

[Can86] J Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6) :679–698, June 1986.

[CB08] F. Cayre and P. Bas. Kerckhoffs-based embedding security classes for woa data hiding. *IEEE Transactions on Information Forensics and Security*, 3(1) :1–15, 2008.

[CCFG16] Jean-François Couchot, Raphaël Couturier, Yousra Ahmed Fadil, and Christophe Guyeux. A second order derivatives based approach for steganography. In *Secrypt 2016, 13th Int. Conf. on Security and Cryptography*, pages 424–431, Lisbon, July 2016.

[CCG+02] Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2 : An opensource tool for symbolic model checking. In Ed Brinksma and Kim Guldstrand Larsen, editors, *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364. Springer, 2002.

[CCG15] Jean-François Couchot, Raphaël Couturier, and Christophe Guyeux. STABYLO : steganography with adaptive, bbs, and binary embedding at low cost. *Annales des Télécommunications*, 70(9-10) :441–449, 2015.

[CCVHG16] Jean-François Couchot, Sylvain Contassot-Vivier, Pierre-Cyrille Héam, and Christophe Guyeux. Random walk in a n-cube without hamiltonian cycle to chaotic pseudorandom number generation : Theoretical and practical considerations. *International Journal of Bifurcation and Chaos*, 2016. Accepted on Oct 2016.

[CDS12] Jean-François Couchot, Karine Deschinkel, and Michel Salomon. Suitability of artificial neural network for MEMS-based flow control. In Julien Bourgeois and Michel de Labachelerie, editors, *dMEMS 2012, Workshop on design, control and software implementation for distributed MEMS*, pages 1–6, Besançon, France, April 2012. IEEE CPS.

[CDS13] Jean-François Couchot, Karine Deschinkel, and Michel Salomon. Active MEMS-based flow control using artificial neural network. *Mechatronics*, 23(7) :898–905, October 2013. Available online.

[CFF05] F. Cayre, C. Fontaine, and T. Furon. Watermarking security : theory and practice. *IEEE Transactions on Signal Processing*, 53(10) :3976–3987, 2005.

[CGH07] Nigel Crook, Wee Jin Goh, and Mohammad Hawarat. Pattern recall in networks of chaotic neurons. *Biosystems*, 87(2-3) :267 – 274, 2007.

[Cha06] Nishanth Chandrasekaran. Verifying convergence of asynchronous iterative algorithms based on lyapunov functions. 2006.

[CHG+14a] Jean-François Couchot, Pierre-Cyrille Héam, Christophe Guyeux, Qianxue Wang, and Jacques Bahi. Pseudorandom number generators with balanced gray codes. In *Secrypt 2014, 11th Int. Conf. on Security and Cryptography*, pages 469–475, Vienna, Austria, August 2014.

[CHG+14b] Jean-François Couchot, Pierre-Cyrille Héam, Christophe Guyeux, Qianxue Wang, and Jacques Bahi. Traversing a n-cube without balanced hamiltonian cycle to generate pseudorandom numbers. 15-th Mons Theoretical Computer Science Days (15e Journées Montoises d'Informatique Théorique), Nancy, France, September 2014.

[Cou10] J.-F. Couchot. Formal Convergence Proof for Discrete Dynamical Systems. Research Report RR2010-03, LIFC - Laboratoire d'Informatique de l'Université de Franche Comté, May 2010.

[CW01] Brian Chen and Gregory W. Wornell. Quantization index modulation methods for digital watermarking and information embedding of multimedia. *Journal of VLSI signal processing systems for signal, image and video technology*, 27(1-2) :7–33, 2001.

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2 :303–314, 1989.

[DD10] Ilker Dalkiran and Kenan Danisman. Artificial neural network based chaotic generator for cryptology. *Turkish Journal of Electrical Engineering and Computer Sciences*, 18(2) :225–240, 2010.

[DP11] Pawel Dabal and Ryszard Pelka. A chaos-based pseudo-random bit generator implemented in fpga device. In *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on*, pages 151–154. IEEE, 2011.

[DP12] Pawel Dabal and Ryszard Pelka. Fpga implementation of chaotic pseudo-random bit generators. In *Mixed Design of Integrated Circuits and Systems (MIXDES), 2012 Proceedings of the 19th International Conference*, pages 260–264. IEEE, 2012.

[DP14] Pawel Dabal and Ryszard Pelka. A study on fast pipelined pseudo-random number generator based on chaotic logistic map. In *Design and Diagnostics of Electronic Circuits Systems, 17th International Symposium on*, pages 195–200, April 2014.

[ERTL11] Andrea Espinel Rojas, Ina Taralova, and René Lozi. NEW ALTERNATE LOZI FUNCTION FOR RANDOM NUMBER GENERATION. In *EPNACS 2011 within ECCS'11*, pages 13–15, Vienne, Austria, September 2011.

[FCCG15] Yousra Ahmed Fadil, Jean-François Couchot, Raphaël Couturier, and Christophe Guyeux. Steganalyzer performances in operational contexts. In *IIH-MSP 2015, 11th Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, pages 429–432, Adelaide, Australia, September 2015.

[FGB11] Nicolas Friot, Christophe Guyeux, and Jacques Bahi. Chaotic iterations for steganography - stego-security and chaos-security. In Javier Lopez and Pierangela Samarati, editors, *SECRYPT'2011, Int. Conf. on Security and Cryptography. SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 218–227, Sevilla, Spain, July 2011. SciTePress.

[FJF11a] Tomás Filler, Jan Judas, and Jessica J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3-2) :920–935, 2011.

[FJF11b] Tomás Filler, Jan Judas, and Jessica J. Fridrich. Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3-2) :920–935, 2011.

[FK12] Jessica J. Fridrich and Jan Kodovský. Steganalysis of lsb replacement using parity-aware features. In Matthias Kirchner and Dipak Ghosal, editors, *Information Hiding - 14th International Conference, IH 2012, Berkeley, CA, USA, May 15-18, 2012, Revised Selected Papers*, volume 7692 of *Lecture Notes in Computer Science*, pages 31–45, Berlin, May 2012. Springer.

[FS07] Lars-Ake Fredlund and Hans Svensson. Mcerlang : a model checker for a distributed functional programming language. *SIGPLAN Not.*, 42 :125–136, 2007.

[FS09] Tomás Feder and Carlos Subi. Nearly tight bounds on the number of hamiltonian circuits of the hypercube and generalizations. *Inf. Process. Lett.*, 109(5) :267–272, February 2009.

[GCS08] E. Goles Ch. and L. Salinas. Comparison between parallel and serial dynamics of boolean networks. *Theoretical Computer Science*, 396(1-3) :247–253, 2008.

[GFB10] Christophe Guyeux, Nicolas Friot, and Jacques M. Bahi. Chaotic iterations versus spread-spectrum : chaos and stego security. In *IIH-MSP'10, 6-th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 208–211, Darmstadt, Germany, October 2010.

[Gra53] Frank Gray. Pulse code communication, 1953. US Patent 2,632,058, March 17 1953,(filed November 13 1947).

[Guy10] Christophe Guyeux. *Le désordre des itérations chaotiques et leur utilité en sécurité informatique*. Thèse de Doctorat, LIFC, Université de Franche-Comté, 13 décembre 2010. Rapporteurs : Pascale Charpin, Directrice de Recherche, INRIA-Rocquencourt ; Eric Filiol, Professeur, ESIEA-Laval ; Pierre Spitéri, Professeur Emérite, IRIT-ENSEEIHT. Examinateurs : Michel de Labachelerie, Directeur de recherche CNRS, Université de Franche-Comté ; Laurent Larger, Professeur, Université de Franche-Comté ; Jean-Claude Miellou, Professeur, Université de Franche-Comté ; Congduc Pham, Professeur, Université de Pau. Directeur : Jacques M. Bahi, Professeur, Université de Franche-Comté.

[Hén76] Michel Hénon. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50(1) :69–77, 1976.

[HF12] Vojtech Holub and Jessica J. Fridrich. Designing steganographic distortion using directional filters. In *WIFS*, pages 234–239. IEEE, 2012.

[HFD14] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014(1), 2014.

[Hol03] Gerard J. Holzmann. *The SPIN Model Checker : Primer and Reference Manual*. Addison-Wesley, Pearson Education, 2003.

[HSW89] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5) :359–366, 1989.

[Ker05] Andrew D. Ker. A general framework for structural steganalysis of lsb replacement. In Mauro Barni, Jordi Herrera-Joancomartí, Stefan Katzenbeisser, and Fernando Pérez-González, editors, *Information Hiding, 7th International Workshop, IH 2005, Barcelona, Spain, June 6-8, 2005, Revised Selected Papers*, volume 3727 of *Lecture Notes in Computer Science*, pages 296–311, Berlin, June 2005. Springer.

[KF11] Jan Kodovský and Jessica Fridrich. Steganalysis in high dimensions : Fusing classifiers built on random subspaces. In *Proc. SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics XIII*, pages 78800L–78800L–13, Bellingham, WA, February 2011. Society of Photo-Optical Instrumentation Engineers.

[KFH12] Jan Kodovský, Jessica J. Fridrich, and Vojtech Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2) :432–444, 2012.

[KPF10] Jan Kodovský, Tomás Pevný, and Jessica J. Fridrich. Modern steganalysis can detect yass. In *Media Forensics and Security*, page 754102, 2010.

[LDX10a] Yantao Li, Shaojiang Deng, and Di Xiao. A novel hash algorithm construction based on chaotic neural network. *Neural Computing and Applications*, pages 1–9, 2010.

[LDX10b] Yantao Li, Shaojiang Deng, and Di Xiao. A novel hash algorithm construction based on chaotic neural network. *Neural Computing and Applications*, pages 1–9, 2010.

[LHH10] Weiqi Luo, Fangjun Huang, and Jiwu Huang. Edge adaptive image steganography based on lsb matching revisited. *IEEE Transactions on Information Forensics and Security*, 5(2) :201–214, June 2010.

[LHS08] Bin Li, Jiwu Huang, and Yun Q. Shi. Textural features based universal steganalysis. In *Proc. SPIE 6819*, page 12, Bellingham, WA, February 2008. Society of Photo-Optical Instrumentation Engineers.

[Lia09] Shiguo Lian. A block cipher based on chaotic neural networks. *Neurocomputing*, 72(4-6) :1296 – 1301, 2009.

[Lor63] E. N. Lorenz. Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, 20 :130–148, March 1963.

[Loz12] R. Lozi. Emergence of randomness from chaos. *I. J. Bifurcation and Chaos*, 22(2), 2012.

[LPW06] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

[LSXC08] Shubo Liu, Jing Sun, Zhengquan Xu, and Zhaohui Cai. An improved chaos-based stream cipher algorithm and its vlsi implementation. In *Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on*, volume 2, pages 191–197. IEEE, 2008.

[LT10] I-Shi Lee and Wen-Hsiang Tsai. A new approach to covert communication via PDF files. *Signal Processing*, 90(2) :557–565, 2010.

[LT14] René Lozi and Ina Taralova. From Chaos to Randomness via Geometric Undersampling. *ESAIM : Proceedings and Surveys*, 46 :177–195, November 2014. Editors : Witold Jarczyk, Daniele Fournier-Prunaret, João Manuel Goncalves Cabral.

[M+76] Robert M May et al. Simple mathematical models with very complicated dynamics. *Nature*, 261(5560) :459–467, 1976.

[MAPS13] Lahcene Merah, Adda ALI-PACHA, and Naima HADJ SAID. Coupling two chaotic systems in order to increasing the security of a communication system-study and real time fpga implementation. *International Journal of Computer Science and Telecommunications*, 4, October 2013.

[MBZ+13] Abhinav S Mansingka, Mohamed L Barakat, M Affan Zidan, Ahmed G Radwan, and Khaled N Salama. Fibonacci-based hardware post-processing for non-autonomous signum hyperchaotic system. In *IT Convergence and Security (ICITCS), 2013 International Conference on*, pages 1–4. IEEE, 2013.

[MCL06] Yaobin Mao, Liu Cao, and Wenbo Liu. Design and fpga implementation of a pseudo-random bit sequence generator using spatiotemporal chaos. In *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, volume 3, pages 2114–2118. IEEE, 2006.

[MU05] M. Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[NK16] A. Mostefaoui N. Khernane, J.-F. Couchot. Maximizing network lifetime in wireless video sensor networks under quality constraints. In *MOBIWAC 2016 : The 14th ACM* International Symposium on Mobility Management and Wireless Access*, November 2016.

[PBF10] Tomás Pevný, Patrick Bas, and Jessica J. Fridrich. Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security*, 5(2) :215–224, 2010.

[PD08] L. Y. POR and B. Delina. Information hiding : A new approach in text steganography. In *7th WSEAS Int. Conf. on APPLIED COMPUTER & APPLIED COMPUTATIONAL SCIENCE*, April 2008.

[PFB10a] Tomás Pevný, Tomás Filler, and Patrick Bas. Break our steganographic system, 2010. Available at http://www.agents.cz/boss/.

[PFB10b] Tomás Pevný, Tomás Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In Rainer Böhme, Philip W. L. Fong, and Reihaneh Safavi-Naini, editors, *Information Hiding - 12th International Conference, IH 2010, Calgary, AB, Canada, June 28-30, 2010, Revised Selected Papers*, volume 6387 of *Lecture Notes in Computer Science*, pages 161–177, Berlin, June 2010. Springer.

[PFPGC06] Luis Perez-Freire, F. Pérez-Gonzalez, and Pedro Comesaña. Secret dither estimation in lattice-quantization data hiding : A set-membership approach. In *Security, Steganography, and Watermarking of Multimedia Contents, San Jose, California*, volume 6072, pages 1–12, Bellingham, WA, January 2006. Society of Photo-Optical Instrumentation Engineers.

[RBBM00] Maria Rifqi, Vincent Berger, and Bernadette Bouchon-Meunier. Discrimination power of measures of comparison. *Fuzzy Sets and Systems*, 110 :189–196, 2000.

[RC81] John P. Robinson and Martin Cohn. Counting sequences. *IEEE Trans. Comput.*, 30(1) :17–23, January 1981.

[RC07] Adrien Richard and Jean-Paul Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics*, 155(18) :2403–2413, 2007.

[RDBM03] Maria Rifqi, Marcin Detyniecki, and Bernadette Bouchon-Meunier. Discrimination power of measures of resemblance. In *IFSA'03*, 2003.

[RDH03] Robby, Matthew B. Dwyer, and John Hatcliff. Bogor : an extensible and highly-modular software model checking framework. In *Proc. of the 11th ACM SIGSOFT Symposium on Foundations of Software Engineering 2003*, pages 267–276. ACM, 2003.

[Rob95] F. Robert. *Les systèmes dynamiques discrets*, volume 19 of *Mathématiques et Applications*. Springer, 1995.

[Rö76] O.E. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5) :397 – 398, 1976.

[SCN09] Rui Miguel Silva, Rui Gustavo Crespo, and Mario Serafim Nunes. Loba128, a lorenz&#45 ;based prng for wireless sensor networks. *Int. J. Commun. Netw. Distrib. Syst.*, 3(4) :301–318, August 2009.

[SJT+04] K. Satish, T. Jayakar, C. Tobin, K. Madhavi, and K. Murali. Chaos based spread spectrum image steganography. *Consumer Electronics, IEEE Transactions on*, 50(2) :587 – 590, May 2004.

[SQW+01] Li Shujun, Li Qi, Li Wenmin, Mou Xuanqin, and Cai Yuanlong. Statistical properties of digital piecewise linear chaotic maps and their roles in cryptography and pseudo-random coding. In *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK*, volume 2260 of *Lecture Notes in Computer Science*, pages 205–221, Berlin, December 2001. Springer.

[Wei97] Carsten Weise. An incremental formal semantics for PROMELA. In *SPIN97, the Third SPIN Workshop*, 1997.

[Wig03] Stephen Wiggins. *Introduction to applied nonlinear dynamical systems and chaos*, volume 2. Springer Science & Business Media, 2003.

[WT08] C. T. Wang and W. H. Tsai. Data hiding in pdf files and applications by imperceivable modifications of pdf object parameters. In *Proceedings of 2008 Conference on Computer Vision, Graphics and Image Processing*, pages 1–6, 2008.

[YK50] G.U. Yule and MG Kendall. An introduction to the theory of. *Statistics*, 1950.

[ZLW05] Linhua Zhang, Xiaofeng Liao, and Xuebing Wang. An image encryption approach based on chaotic maps. *Chaos, Solitons & Fractals*, 24(3) :759 – 765, 2005.

[ZS04] A. J. van Zanten and I. N. Suparta. Totally balanced and exponentially balanced gray codes. *Discrete Analysis and Operational Research*, 11 :81–98, 2004.

TABLE DES FIGURES

LISTE DES TABLES

Résumé :

Grâce à leur concision, les modèle discrets permettent d'appréhender des problèmes informatiques qui ne seraient parfois pas traitables autrement. Les systèmes dynamiques discrets s'intègrent dans cette thématique. Dans cette habilitation, nous montrerons tout d'abord des contributions concernant la convergence, la preuve de convergence et un nouveau mode opératoire de tels systèmes. Nous présenterons ensuite un ensemble d'avancées autour des fonctions dont les itérations peuvent être chaotiques. Particulièrement, plusieurs méthodes permettant d'obtenir de telles fonctions seront proposées, dont une basée sur les codes de Gray, fournissant, en plus une, chaîne de Markov doublement stochastique. Grâce à cette dernière, nous pourrons notamment engendrer une grande famille de générateurs de nombres pseudo-aléatoires (PRNG). Des contributions théoriques et pratiques autour de ces PRNGs seront mises en avant. La thématique de masquage d'information (déjà présente dans l'équipe) a été renforcée et des avancées significatives sur ce sujet seront présentées. Des instances de ces algorithmes seront formalisées en sélectionnant les fonctions à itérer pour garantir une robustesse élevée. Finalement, nous montrerons qu'on peut construire de nouvelles fonctions de distorsion utilisables en masquage d'information à l'aide de méthodes d'analyse par gradient mais discret cette fois encore.

Mots-clés : systèmes dynamiques discrets, générateurs de nombres pseudo-aléatoires, masquage d'information.

Abstract:

Thanks to its conciseness, a discrete model may allow to reason with problems that might not be handled without such formalism. Discrete dynamical systems belong to this computer science area. In this authorization to direct researches manuscript, we firstly present contributions on convergence, convergence proof, and a new iteration scheme of such systems. Then we offer contributions about functions whose iterations can be chaotic. In particular, we present a set of methods leading to such functions. One of them, built over Gray codes, allows to obtain a Markov chain that is doubly stochastic. This last method permits to produce a large number of Pseudorandom Number Generators (PRNG). Theoretical and practical contributions are presented in this field. The Information hiding area has been strengthened in this manuscript and some contributions are thus presented. Instances of such algorithms are given according to functions that can achieve a large robustness. Finally, we have proposed a new method to build distortion functions that can be embedded in information hiding schemes with analysis gradient but expressed in a discrete way.

Keywords: discrete dynamical systems, pseudorandom number generators, information hiding.

SPIM

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex

■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ