Automates Cellulaires; Fonctions Booléennes

Jean-Baptiste Yunès

À propos d'automates cellulaires

suivi par

Des fonctions Booléennes

Mémoire constitué en vue de l'obtention d'une

Habilitation à Diriger des Recherches

devant le jury constitué de :

Emmanuel Chailloux Professeur à Paris 6
Guy Cousineau Professeur à Paris 7
Patrick Greussay Professeur à Paris 8
Serge Grigorieff Professeur à Paris 7
Jean Mairesse Directeur de Recherche à Paris 7 (Rapporteur)
Jacques Mazoyer Professeur à l'ENS-Lyon (Rapporteur)
Jean Francis Michon Professeur à Rouen
Jean-Marie Rifflet Professeur à Paris 7

et avec l'aide de :

Hiroshi Umeo Professor, Osaka, Japan (Rapporteur)
Thomas Worsch Assistant Professor, Karlruhe, Germany (Rapporteur)

par Jean-Baptiste Yunès

le 12 décembre 2007

Table des matières

Remerciements[1]

Ainsi la machine à calculer que l'on surmène, s'embrouillant soudainement dans ses fiches perforées, rejoint dans la mesure de ses moyens le délire poétique, fait un saut du règne de la compilation à celui de la création.
(G.-E. Debord, Œuvres complètes, 2006)

Le professeur Jean-Marie Rifflet sait avec quel respect je le considère. Outre le fait qu'il est un enseignant tout à fait hors-pair et que les souvenirs d'étudiant qu'il m'a laissés sont magnifiques, je me délecte désormais chaque jour de l'amitié qu'il me porte et dont j'espère être digne. Merci Jean-Marie. Vraiment.

Pour remercier le professeur Serge Grigorieff il me faudrait être pour quelqu'un d'autre ce qu'il a été pour moi, je crois qu'il en serait heureux. Je ne sais pas avec quelle «magie» il a su tirer, de moi, le meilleur, mais j'espère bien être moi aussi doté de ce don. Je ne désespère pas non plus que notre amitié puisse être conservée telle quelle pour longtemps encore.

L'amitié qui me lie au professeur Jean-Francis Michon, bien que plus récente n'est certainement pas moins profonde, de toute façon chacun sait que les amitiés ne se comparent pas. Nous avons passé, ci et là, quelques merveilleux moments à discuter et travailler. J'en redemande bien volontiers.

Après une longue hésitation j'ai demandé au professeur Guy Cousineau de participer à ce jury (il occupe aujourd'hui une position qui induit nécessairement la gestion d'un emploi du temps infernal), mais lui n'a pas hésité, il a accepté dans l'instant. C'est un grand honneur pour moi. Moi qui ai commencé mes études à Paris 7, je suis fier qu'il dirige maintenant l'établissement auquel je suis très attaché.

[1] L'ordre des remerciements n'est pas chargé sémantiquement. En LaTeX on aurait pu avoir à sa disposition une macro comme `\parthanks`, je l'aurais utilisée !

Le professeur Emmanuel Chailloux a été étonné (je le sais humble) de ma demande de participation à ce jury ; et pourtant il sait très bien que nous avons tous deux des passions communes. Je crois que nous nous comprenons très bien.

Nous nous sommes longtemps poliment croisés mais le hasard d'une refonte cosmétique des équipes du laboratoire m'a conduit vers Jean Mairesse, aujourd'hui Directeur de Recherche. Quel ne fût pas mon étonnement de constater qu'il portait un intérêt non feint à mes préoccupations, il faisait alors et de toute évidence preuve d'une véritable curiosité. C'est à mes yeux l'une des plus belles qualités et je ne peux que respecter ceux qui la possèdent. Je le remercie aussi d'avoir accepté d'être rapporteur de mon dossier.

Je remercie sincèrement le professeur Jacques Mazoyer qui malgré nos contacts irréguliers marque à mon égard un intérêt qui m'honore. Il a accepté de fournir un rapport sur mes travaux, qu'il en soit particulièrement remercié.

À chaque fois que je croise le professeur Patrick Greussay (c'est arrivé trop peu de fois) je regrette ensuite de ne pas avoir provoqué une telle rencontre plus souvent. Sa vision de la discipline est teintée d'un pragmatisme très cher à mes yeux.

Je tiens à remercier les deux autres rapporteurs : le professeur Hiroshi Umeo avec qui nous passons de très agréables moment lors de nos retrouvailles car notre petite faiblesse commune pour le Firing Squad nous rapproche de façon particulière ; et le Docteur Thomas Worsch, nos discussions à bâtons rompus dans les conférences où nous nous croisons parfois me sont précieuses.

Dany, Antoine, Luc, Juliette : j'aime tant lorsque nous nous tenons la main, dans les bois, sur le chemin de l'école... J'aime tant lorsque nous rions à gorge déployée, goûtant notre bonheur... Je vous aime.

Chapitre 1

Introduction

Ce document se propose de présenter divers travaux que j'ai pu entreprendre depuis mon entrée dans le monde de la recherche.

Tout d'abord, sont issus de mon éveil à la recherche, une série de travaux autour des automates cellulaires et en particulier un fameux problème inverse, le « Firing Squad ». Bien qu'étant l'objet de très nombreuses publications, sa richesse n'est pas épuisée, ce que j'espère bien montrer à travers ce document. Ce problème est à mes yeux représentatif des méthodes de calcul que nous auront à mettre en place dans le futur et il est, assurément, bien loin d'être seulement un amusement.

C'est par le biais d'une tentative d'acquisition d'une culture en cryptographie que me sont apparues les fonctions Booléennes. L'exploration de leur monde a donné naissance à quelques travaux qui constituent donc la deuxième partie de ce document. Ces travaux étant plus récents que ceux de la première partie, le volume du texte en est d'autant réduit, pour l'instant seulement.

Si les deux parties précédentes sont apparemment distinctes, je ne désespère pas, un jour, de concilier formellement ces différentes approches de l'informatique. Je possède encore deux autres marottes : les systèmes d'exploitation et la programmation. Si mes acquis en ces deux dernières disciplines n'ont pas donné lieu à des publications scientifiques ordinaires c'est, non seulement parce qu'on ne peut décemment se diviser à l'envie, mais aussi parce qu'elles manquent de formalisation et que moi, pas plus qu'un autre, ne sait encore comment les aborder efficacement ainsi. Certes ce manque peut être nuancé pour la programmation, en particulier pour le fonctionnel qui ne manque pas d'assise, mais c'est évidemment moins clair pour l'impératif. En ce qui

concerne le système, il n'existe même pas de définition satisfaisante de ce qu'est un système. Mais on doit pouvoir déceler, en filigrane, à la lecture des différents ouvrages auxquels j'ai participé, l'immense intérêt que j'y porte.

Il n'y a pour moi, dans tout cela, aucune fracture, aucune rupture, bien au contraire. S'il y a bien longtemps que les ordinateurs de toutes sortes me sont familiers, la discipline me cache encore bien des secrets. Et pourtant, je pressens vraiment l'existence d'une unité, d'un fil conducteur qui nous promènerait continûment de nos pensées rationnelles aux algorithmes, des algorithmes aux programmes, et des programmes aux machines. Je ne prétend pas pouvoir en être le seul auteur, j'espère simplement y contribuer, humblement, par la communication de mon savoir accumulé ; à mon tour, comme l'ont fait mes prédécesseurs.

Chapitre 2

Travaux sur les automates cellulaires

2.1 Des automates cellulaires au firing squad

2.1.1 Un modèle de calcul

C'est à John von Neumann[1], célèbre mathématicien autant que physicien, que l'on doit ce modèle de calcul. Il lui permit d'abstraire mathématiquement un problème d'auto-reproduction qui lui tenait tête, la modélisation qui s'ensuivit lui permit de le résoudre. Sa solution est un automate cellulaire universel auto-reproducteur plan à 29 états.

Sans doute le charme de ce modèle de calcul est-il dû à cette histoire originelle, l'auto-reproduction que l'on conçoit aisément comme un mécanisme essentiel de la vie est fascinante. On doit certainement à Arthur Burks (voir [9]) d'avoir produit un monument compilant les problèmes les plus importants de la théorie des automates cellulaires de cette époque. Ensuite ce sont sans doute les physiciens et les biologistes qui se sont intéressés à cet objet, car il leur permettait de modéliser de façon adéquate nombre de leurs problèmes. Mais aujourd'hui il est facile d'observer que de nombreuses disciplines s'y intéressent, et que de fructueuses relations s'établissent encore

[1]John von Neumann (1903–1957) était un scientifique très prolifique et de première importance. Il a contribué de façon très significative à de nombreux domaines scientifiques. Il a donné son nom à « l'architecture de von Neumann » qui constitue encore aujourd'hui l'ossature des ordinateurs. Il a en outre contribué à la création de l'EDVAC, l'un des premiers ordinateurs électroniques américains.

entre elles. Il existe non seulement de très nombreux articles scientifiques à propos des automates cellulaires mais aussi des livres. Le lecteur curieux pourra par exemple se reporter aux ouvrages de Arthur Burks ([9]), Edgar Codd ([12]), Stephen Wolfram ([104, 105]) ou encore Max Garzon ([18]). Les actes de congrès sont aussi très nombreux.

Les problématiques sont nombreuses et toutes très vivantes, mais on peut citer les principales : l'auto-reproduction, l'universalité, la vie artificielle, la résistance aux pannes, la reconnaissance de langages, la réversibilité, le firing squad, le jeu de la vie, la classification, les propriétés limites, la dynamique symbolique, la topologie, la complexité, l'architecture...

La simplicité de ce modèle est telle qu'une rapide description permet à tout un chacun de saisir instantanément de quoi il s'agit ; ce qui n'est pas le cas d'autres modèles.

> Un *automate cellulaire* peut être considéré comme un ensemble de composants élémentaires identiques appelés *cellules* et connectées les unes aux autres. À un instant donné chaque cellule se trouve dans un certain *état* lequel varie au cours du temps de la façon suivante : les états de toutes les cellules sont modifiés en même temps à chaque top d'une horloge commune, en fonction de l'état de chacune de ses *voisines* qui lui sont connectées, et selon des *règles* prédéfinies.

Prenons par exemple le cas d'un automate cellulaire composé d'une « grille » de cellules pouvant être dans l'état `1` ou `0` et utilisant la *fonction de transition* (ensemble des règles) suivante : si une cellule est dans l'état `1` elle passe à `0` si moins de deux ou plus de trois de ses voisins sont à `1` sinon elle conserve son état, et dans l'état `0` elle passe à `1` si exactement trois voisines sont à `1` et conserve son état sinon. L'évolution de l'automate est alors celui que l'on peut observer dans la figure 2.1 (on imagine qu'au départ le reste de la grille est plein de `0`), pour des raisons de commodité, nous avons pris soin de colorier les cellules qui contiennent un `1` en noir.

Cet automate cellulaire est plus connu sous le nom de « Jeu de la vie » et est dû à John H. Conway (voir [6]). Nous laissons le lecteur découvrir par lui-même la richesse de cet objet (une visite sur le web fera son bonheur d'expérimentateur).

Bien entendu, saisir la « richesse » (la puissance) d'un tel modèle est d'une toute autre nature ; mais l'observation du plus simple d'entre eux est une affaire *a priori* ordinaire et qui pose beaucoup de questions. Il suffit

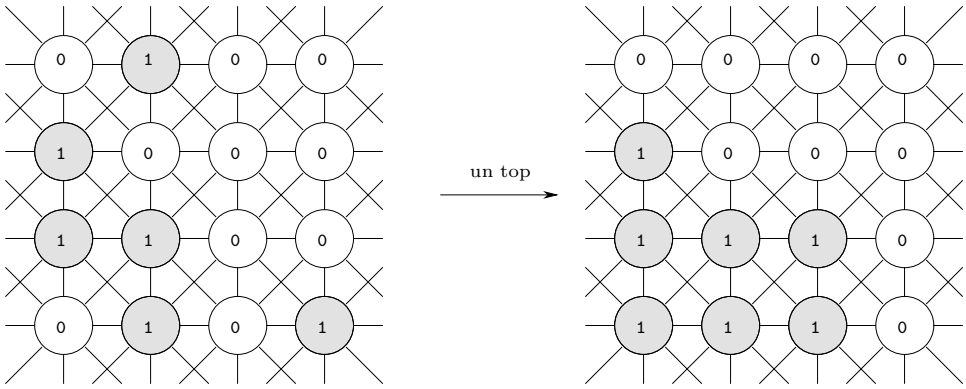


FIG. 2.1 – Le jeu de la vie

de constater que le nombre de chercheurs actifs dans l'exploration des 256 automates classifiés par Stephen Wolfram est important, et que pourtant les questions qu'ils se posent sont si simples à comprendre.

Ce modèle est si expressif que la représentation de l'espace-temps est généralement « parlante » ; du moins pour les petites dimensions et les tailles raisonnables. On y « voit » les choses. Ce document présentera de nombreux diagrammes espace-temps de ces merveilleuses machines, et tout lecteur y verra de jolis dessins dont certaines (ir-)régularités seront d'elles-mêmes signifiantes.

L'exemple suivant, que l'on peut observer en figure 2.2, est ce que l'on appelle un diagramme espace-temps ou encore diagramme d'évolution d'un automate cellulaire constitué d'une simple ligne de cellules. La première ligne représente l'état des différents cellules au départ du calcul, la ligne suivante l'état des cellules après le premier top d'horloge (et application de la fonction de transition), etc.

2.1.2 Un modèle au parallélisme massif

La tâche du scientifique est d'aller au-delà de ces simples visions ou interprétations communes, il se doit de formaliser le tout afin de décrire précisément ce dont il s'agit, pour prédire ou comprendre (ou du moins de tenter de).

Les problèmes étudiés peuvent être rangés dans deux grandes classes. La classe des « forward problems » constituée des problèmes où la question posée

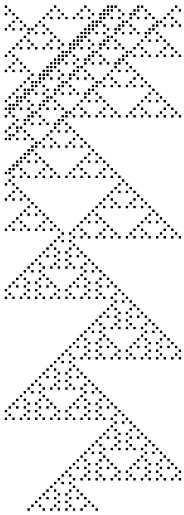


FIG. 2.2 – Un automate cellulaire linéaire et son évolution

est essentiellement de comprendre ce qu'une machine donnée réalise ; c'est l'occupation principale de la dynamique discrète ou symbolique. En second, donc, on trouve la classe des « inverse problems », où la problématique se situe *a contrario* dans la construction d'une machine devant réaliser un objectif donné. Cette problématique est proche de la démarche d'ingénierie ; et en informatique est en grande partie liée à la programmation, cet art (ainsi que le définit Donald E. Knuth, voir [40]) qui consiste à concevoir des algorithmes puis écrire des programmes les réalisant.

Concevoir des algorithmes sur un modèle comme les automates cellulaires n'est pas chose aisée. Si la compréhension d'une machine RAM n'est pas aussi accessible que celle d'un automate cellulaire, inversement la programmation d'une machine RAM est généralement considérée comme plus facile que celle d'un automate cellulaire. La probable raison est que l'esprit humain est plus généralement enclin à raisonner séquentiellement et tout particulièrement en informatique.

Le procédé employé est nommé décomposition (ou encore factorisation) : le but fixé est obtenu par décomposition du problème puis enchaînement en séquence de tâches élémentaires. Aujourd'hui un second type de décomposition est fréquemment utilisé, la décomposition orientée objet, mais nous savons que cette technique n'est pas encore passée dans les mœurs, loin s'en faut,

et peut être n'est-ce pas seulement dû à sa récente diffusion. On notera que même dans le monde orienté objet la séquentialité est très présente : l'échange des messages entre objets est fortement séquentialisé.

Or le modèle qui nous intéresse est lui parallèle : il est intrinsèquement constitué de composants qui calculent en parallèle et de façon synchrone (sans synchronisation on tombe dans le monde du distribué : un tout autre monde). Et l'on constate qu'aujourd'hui la programmation de telles machines est souvent réduite à deux choses : conception de tâches élémentaires dont le calcul est lui parallélisé et enchaînement séquentiel de ces tâches (la création d'un calcul entièrement parallélisé est souvent trop ardu). Mais l'enchaînement de ces tâches nécessite alors des mécanismes de synchronisation. Il en existe de toutes sortes mais l'essentiel d'une synchronisation est d'obtenir que chaque élément soit prêt en même temps que tous les autres. Et le problème considéré comme représentatif d'un tel comportement est celui du « Firing Squad ».

Toutefois l'intérêt du Firing Squad, ne réside pas tant dans l'existence de nombreuses solutions que dans la compréhension des mécanismes sous-jacents. En effet, depuis les solutions de Minsky-McCarthy et Waksman-Balzer ce problème ne présente plus véritablement d'intérêt dans le cadre de la synchronisation ni même celui de la minimisation. Mais, la compréhension des algorithmes utilisés, la façon dont on peut les implémenter et le coût de ses implémentations est un point extrêmement important.

Bien que par le passé les tentatives de construction, et surtout de commercialisation de machines massivement parallèles ont échoué nous ne pouvons en déduire aussi facilement que ce modèle est voué à l'échec. John von Neumann notait en 1948 (lire les actes du Hixon Symposium - [101]) qu'il y avait deux obstacles majeurs sur la route qui mène à une puissance calculatoire équivalente à celle obtenue par les organismes vivants : la taille des composants élémentaires et le manque de théorie adéquate. En ce qui concerne le premier de ces arguments, la question est sans doute en passe d'être réglée, la finesse de gravure du Silicium est aujourd'hui telle que des processeurs ordinaires contiennent plusieurs millions de transistors, et que si nous sommes loin du nombre de neurones d'un cerveau humain, la mise en réseau de milliers de ces processeurs est possible et porte la complexité à un niveau tout à fait honorable[2].

De ce point de vue la plus célèbre des tentatives a été sans aucun doute

[2] 10^{11} neurones estimés dans un cerveau humain. La dernière machine massivement parallèle d'IBM est livrée avec 32.000 processeurs de 5.000.000 de transistors chacun.

la « Connection Machine » qui pouvait posséder jusqu'à 65.000 processeurs ; elle a été initialiement conçue au MIT et dans les années 80 par Daniel Hillis (élève de Marvin Minsky, voir [34, 35]). Il est important de souligner que Richard Feynman[3] lui-même y porta un grand intérêt et travailla même un temps sur sa conception (voir [36]). Sinon, que dire des récentes tentatives comme celle d'IBM de construire une machine, de nom Blue Gene/P, extensible à presque 300.000 unités de calcul ?

Pour le second point noté par von Neumann, le problème est encore entier, car comme il le signifiait lui même nous n'avons pas de vrai retour d'expérience sur ce type d'objet, et que nous ne pouvons certainement pas tout extrapoler. Patrick Greussay (voir [27]) est lui aussi partisan de l'existence d'un saut quantitatif. Et comme Léonid Levin (voir [49]) qui débute ses notes de cours sur les modèles de calcul par la présentation des automates cellulaires, nous pensons que ce modèle est adéquat pour rendre compte des phénomènes intrinsèques au parallélisme vraiment massif.

Il s'agit donc ici d'une étude plus tournée vers une informatique dont l'algorithmique s'exprimerait géométriquement ou algébriquement sur automates cellulaires que sur la synchronisation ici devenue simple prétexte, car nous pensons que, non seulement ce modèle est d'importance et qu'il révélera à l'avenir toute son utilité, mais aussi que la géométrie et l'algèbre seront au cœur de la structure de ses calculs (lire [16, 20, 27, 36, 49, 101, 105]).

2.1.3 Le Firing Squad

L'histoire remonte à 1957, et l'on attribue à John Myhill la première formulation du problème. Voilà bien un argument de plus pour affirmer qu'il s'agit d'un problème essentiel : il s'est immédiatement imposé. Mais c'est Edward Moore qui publiera le premier article y faisant explicitement référence (voir [57]). De façon simple on peut exprimer le problème de la façon suivante :

> Il s'agit de concevoir un automate cellulaire permettant d'obtenir à partir d'une configuration où une seule cellule est active, une configuration dans laquelle toutes les cellules sont après un certain

[3]Richard Feynman (1918–1988) est un des grands physiciens du 20° siècle qui a beaucoup contribué à l'essor de la mécanique quantique mais il est aussi très connu pour son charisme et son extraordinaire don pour la pédagogie qui lui valurent son surnom de « Great Explainer ».

laps de temps, toutes dans un même état ; et que cet état ne soit jamais apparu avant.

Et depuis que d'articles scientifiques à ce sujet !

2.1.3.1 Une solution : Minsky/McCarthy

On doit à Marvin Minsky et John McMarthy (voir [56]) la description du premier algorithme destiné à résoudre ce problème. La technique employée est désormais bien connue, il s'agit du « diviser pour mieux régner » (« divide and conquer » dans le texte) : pour résoudre le problème coupons l'espace en deux parties égales et résolvons-le dans les deux sous-parties en parallèle (*i.e.* en même temps et de façon indépendante). En procédant ainsi dans un espace discret et fini, il devient évident qu'un tel procédé convergera : lorsque que l'on ne pourra plus couper c'est que le processus aura atteint son but. Reste une question : comment couper en deux parties égales ? Le procédé employé est celui illustré par la figure 2.3(a) où les traits représentent la trace idéalisée de messages envoyés entre les cellules.

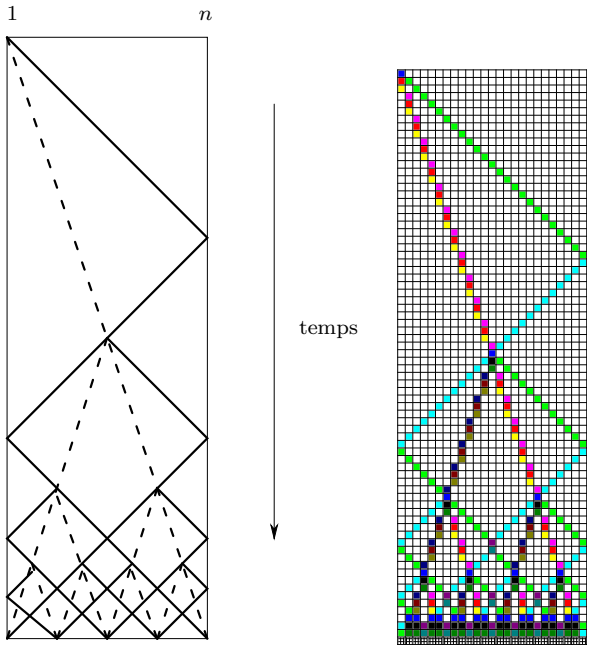
L'implémentation (c'est le nom que l'on donne à un programme réalisant un algorithme sur une machine) de cette technique est du domaine du folklore, ainsi que le web en témoigne : on y trouvera facilement de telles implémentations ; la figure 2.3(b) en offre une possible en 15 états. Elle synchronise toute ligne de longueur n en $3n + O(log(n))$ pas de calcul et son travail (c'est-à-dire le nombre de cellules du diagramme espace-temps « où il se passe quelque chose ») est de l'ordre de $n.log(n)$ (nous définirons ces notions plus tard, mais chacun peut s'en faire une idée à l'observation de la figure).

2.1.4 Plus (ou moins)... la complexité

Comme bien souvent en informatique, se pose alors la question de savoir s'il est possible de faire mieux. Mieux ? Mais en quel sens ? En tout sens possible est la réponse.

2.1.4.1 Complexité en temps (de calcul)

Tout d'abord on peut considérer la complexité en temps, c'est-à-dire le temps que met la machine, en partant de la donnée initiale, à exécuter le programme jusqu'à l'obtention du but recherché. Cette question de « rapidité » (efficacité en temps) est habituellement la première qui vient à l'esprit



(a) (b) $l = 25$

FIG. 2.3 – Le « divide and conquer »

des informaticiens contemporains (les machines modernes offrent de telles capacités de stockage que c'est bien le temps de calcul qui importe le plus).

Dans les automates cellulaires le comptage est simple, il suffit de compter les tops d'horloge ; ce qui est bien plus simple que dans les cas plus classiques où les différentes instructions sont à distinguer (certaines sont considérées, à juste titre, comme plus coûteuses que d'autres). Dans le monde du massivement parallèle, les processeurs sont élémentaires, ils ne font qu'appliquer une fonction de transition dont la simplicité est telle qu'une représentation sous forme de table est la plus adaptée, d'où le coût uniforme des pas de calcul.

Dans la suite on notera par $\mathcal{T}(n)$ le temps mis par une *solution* donnée pour synchroniser une ligne de longueur n.

Dans le cadre du Firing Squad, la première question qu'il est naturel de se poser après qu'une solution ait été trouvée est de déterminer le temps minimal en lequel il est possible de synchroniser. Il n'est pas aussi facile qu'il pourrait paraître de répondre à cette question. En effet, bien que sur une

ligne la réponse et sa preuve soit simple ($2n-2$ pas de calcul[4] pour une ligne de longueur n), dans le cas général c'est moins évident ; et en particulier sur les graphes (voir [28, 39, 42]). Des travaux récents de Kobayashi (voir [42, 45, 21, 22]) suggèrent qu'il existe des familles uniformes de graphes pour lesquelles il n'existe pas de solution en temps minimal ; le temps minimal étant déterminé par le plus petit des temps obtenus par l'ensemble des solutions synchronisantes de cette famille de graphes. Ce problème est très intéressant, car il suggère pour la première fois qu'il peut être inévitable de chercher des solutions non optimales en temps.

2.1.4.2 Complexité en taille (de programme)

Quel est le programme le plus petit (constitué de plus petit nombre d'instructions élémentaires) qui permet de résoudre le problème ? Une telle mesure est appelée « complexité de Kolmogorov ». C'est une question importante pour les théoriciens, quoiqu'en pratique on puisse se la poser dans des environnements très restreints comme l'embarqué. Peu de problème inverses ont été étudiés sous l'angle de l'informatique théorique, c'est pourquoi les études autour de cette question dans les automates cellulaires sont rares. Dans le cadre du Firing Squad on notera qu'ici et là sont mentionnés la taille des règles effectivement utilisées pour synchroniser. Le record en la matière est la solution de Umeo (voir [92]) qui n'utilise que 78 règles de transition (ou instructions si l'on préfère) ; nombre très nettement plus petit que les valeurs habituellement observées (plus d'une centaine).

On notera donc dans la suite par $\mathcal{K}(\text{FSSP})$, la complexité de Kolmogorov du problème. On notera par $\mathcal{P}(\text{solution})$ la taille du programme implémentant une *solution* donnée. On a la relation

$$\forall \text{solution}, \mathcal{K}(\text{FSSP}) \leq \mathcal{P}(\text{solution})$$

Avec la solution de Umeo, nous savons que $\mathcal{K}(\text{FSSP}) \leq 78$ (voir [92]).

[4] On trouvera parfois dans la littérature que le temps minimal est $2n-1$, comme toujours cela dépend de quoi l'on parle. Il y a toujours au moins $2n-2$ pas de calculs nécessaires pour synchroniser ; mais si l'on parle du temps auquel la synchronisation se produit, tout change. En partant du temps 0, la synchronisation peut s'obtenir au temps $2n-2$, sinon en partant de 1, c'est au temps $2n-1$.

2.1.4.3 Complexité en taille (de mémoire)

De combien de mémoire a-t-on besoin pour résoudre le problème ? C'est une question qui se pose de façon moins prégnante aujourd'hui mais elle fût tout à fait cruciale lors de l'essor de la technologie informatique : les ordinateurs d'alors n'avait que (très) peu de mémoire disponible.

Dans le cadre du Firing Squad, cette mesure correspond au cardinal de l'ensemble des états sur lesquels agit la fonction de transition ; et c'est habituellement cette mesure qui est considérée comme de première importance[5]. Dans la suite, on notera simplement par $|Q|$ la cardinalité de l'ensemble Q des états de l'automate.

Il a été prouvé par Balzer (voir [4]) qu'il n'existe pas de solution à 4 états en temps minimal au problème du Firing Squad. Ainsi donc, nous savons que la plus petite taille d'automate vérifie la propriété suivante

$$4 < |Q| \leq 6$$

Car l'on sait désormais depuis Mazoyer (voir [51]) qu'il existe une solution en 6 états à ce problème.

La preuve de l'inexistence de solution à 4 états n'est pas formelle, elle est basée sur un algorithme de recherche exhaustive. Plusieurs auteurs ont réécrit un tel programme, Yunès (voir [107]), Sanders (voir [74]). Sans doute une preuve formelle serait-elle instructive, et permettrait peut-être de trouver des heuristiques pertinentes pour l'exhaustion des solutions à 5 états.

On notera que le résultat ne concerne que les solutions à 4 états en temps minimal et qui synchroniserait toutes les lignes. En ce qui concerne les temps non minimaux et/ou un sous-ensemble infini des longueurs possibles de ligne c'est moins clair. En effet, des travaux en cours (Umeo et Yunès de façon indépendante) font apparaître qu'il existe plusieurs solutions à 4 états permettant de synchroniser un ensemble infini de lignes : un sous-ensemble de $\mathbb{N}$ (voir § 2.1.9.2).

2.1.4.4 Complexité en énergie

Une autre mesure qui pourrait nous être utile afin de classifier les solutions existantes est la notion de travail. Introduite par Vollmar (voir [100]), elle

[5]Une taille de mémoire physique s'exprime habituellement en nombre de bits, qui est le logarithme en base 2 du nombre d'états. Nous nous affranchissons ici de cette question de représentation qui introduit de façon générale un logarithme (sauf pour la base unaire). Nous compterons donc simplement les états...

correspond intuitivement à l'énergie consommée par l'automate. On mesure donc habituellement le nombre de messages transmis pendant le calcul en considérant qu'un état quiescent n'envoie pas de message à ses voisins. Vollmar a montré que pour résoudre le Firing Squad il est nécessaire d'échanger un nombre de messages de l'ordre de $n.log(n)$ pour une ligne de n automates.

Dans la suite nous noterons la complexité d'une *solution* donnée par $W_{\text{solution}}(n)$. En général les solutions ont une complexité $W(n) = \mathcal{O}(n^2)$ ou $W(n) = \mathcal{O}(n.log(n))$. Il s'agit là des deux complexités rencontrées jusqu'à aujourd'hui, car des résultats récents semblent laisser penser qu'il existe peut être une complexité de travail fractale, *i.e.* $n.log(n) \leq W(n) \leq n^2$.

2.1.5 La classification des solutions

La classification est un exercice difficile, d'abord parce qu'il faut avoir un bon critère discriminant et ensuite parce que ce critère doît être effectif. C'est tout le problème de la classification de Wolfram (voir [104, 105]), source de nombreuses critiques mais qui a le mérite d'exister. Sa classification des automates cellulaires élémentaires est basée sur l'observation, pratique courante dans les sciences expérimentales ou naturalistes, mais considérée comme incongrue en mathématiques. Ici nous essayons d'établir une liste de critères, si possible exclusifs les uns des autres et qui puissent permettre de mettre un peu d'ordre dans les solutions au problème de la synchronisation.

Bien entendu et de façon très traditionnelle en informatique, nous trouvons d'abord la complexité, parce qu'elle représente quelque chose en termes de machines (rapidité, taille, énergie). Ensuite, nous proposons, comme l'a déjà tenté Umeo (voir [89]), des critères permettant de classer les algorithmes. Ainsi lorsque le zoo des solutions sera encore un peu plus peuplé, pourrons-nous, peut-être, établir des liens raisonnés entre les algorithmes géométriques et leur implémentation, et par là suggérer un prototype de langage de programmation et son compilateur...

Les critères de classification des algorithmes sont :

- les critères de complexité déjà évoqués ;
- le type de procédé employé. On en trouvera deux : l'itération et la récursion. L'itération est à considérer par opposition à la récursion où le procédé est d'imbriquer le développement de l'algorithme dans lui-même à l'aide d'un changement d'échelle et d'une possible symétrie. Sans symétrie, on dira qu'il s'agit d'une récursion unidirectionnelle sinon bidirectionnelle ;

- la symétrie. Ici encore, on en trouvera de deux sortes. La symétrie simple où la fonction de transition vérifie la propriété

$$\forall x, y, z \in Q,\ \delta(x, y, z) = \delta(z, y, x)$$

La double symétrie qui est une symétrie composée avec une bijection entre deux sous-ensembles distincts des états, *i.e.*
$Q = Q' \bigcup Q'' \bigcup \{q, f\}$, et que $\exists \varphi$ bijective : $Q' \rightarrow Q''$ telle que

$$\forall x, y, z \in Q,\ \varphi(\delta(x, y, z)) = \varphi(\delta(\varphi(z), \varphi(y), \varphi(x)))$$

On notera qu'il n'est pas nécessaire que les ensemble Q' et Q'' soient disjoints, ainsi le cas où ils sont égaux et que la bijection est simplement la fonction identité, on retombe dans le cas de la simple symétrie ;
- le nombre de type de signaux. En ce cas, on distingue le cas d'une finitude de signaux différents (généralement peu) et l'infinité de signaux. Pour cette dernière, on peut se demander comment un nombre infini de signaux peuvent se placer dans un espace fini discret. L'observation des figures 2.5–2.9, que l'on verra plus loin, éclairera le lecteur, mais à ce point, il suffit d'imaginer cette infinité comme *potentielle* : on verra de plus en plus signaux alors que la longueur de ligne croît.

2.1.6 Les solutions en temps minimal

On appelle solution en temps minimal toute solution qui lorsqu'elle synchronise une ligne de longueur n le fait en temps $\mathcal{T}(n) = 2n-2$. La recherche de solution en temps minimal est une quête quasi-systématique depuis l'origine du problème, car la conception d'un algorithme capable de le résoudre constitue généralement un véritable tour de force.

2.1.6.1 Goto

Cette solution a les caractéristiques suivantes :
$\mathcal{T}(n) = 2n - 2$, $\mathcal{P}(\text{goto}) \leq 10^9$, $|Q| \leq 10^6$, $W(n) = \mathcal{O}(n.log(n))$, itérative, non symétrique, finitude de type signaux.

La publication restreinte du document de Eiichi Goto[6] (il s'agit de notes de cours) (voir [24]) a contribué à ce que cette très intéressante solution

[6]Le professeur Goto s'est éteint récemment – en Juin 2005. C'était un chercheur extrêmement important, très prolifique et qui a contribué, entre autres, de façon essentielle à l'essor des premiers ordinateurs Japonais.

reste oubliée pendant de très nombreuses années. On doit à Hiroshi Umeo (voir [87]) d'avoir exhumé l'algorithme, ainsi qu'à Jacques Mazoyer (voir [53]) d'avoir rédigé la preuve formelle de son fonctionnement ; deux publications malheureusement elles aussi restées confidentielles (en particulier la seconde). Sa solution reste particulièrement astucieuse, puisqu'une seule autre connue lui ressemble (voir Gerken).

Le principe est de décomposer la ligne en une suite de segments dont les longueurs constituent la suite des puissances de 2. Ensuite une construction permet de lancer dans chacun de ces segments des solutions en temps 3n (donc type Minsky-McCarthy) ; le tout de sorte que le temps global obtenu soit minimal. Le point extrêmement important est que ce schéma n'est pas récursif, la fonction de synchronisation ne s'appelle pas elle-même. De plus, pour réaliser la synchronisation elle utilise des solutions non minimales en temps !

Le nombre important d'états de la solution originale est dû à la structuration en produit cartésien de l'ensemble des états : l'automate cellulaire est vu comme une sorte de superposition d'automates cellulaires plus simples. Cependant, des travaux personnels en cours autour de cette question montrent qu'il est possible d'implémenter cet algorithme (où un proche cousin) à peu de frais (quelques dizaines d'états seulement, voir [119]).

2.1.6.2 Waksman/Balzer

La solution de Waksman (voir [103]) possède les caractéristiques suivantes :
$\mathcal{T}(n) = 2n - 2$, $\mathcal{P}(\text{waksman}) = 202$, $|Q| = 16$, $W(n) = \mathcal{O}(n^2)$, récursive bidirectionnelle, symétrique double, infinité de signaux. Les caractéristiques originelles ont été revues et corrigées par Umeo (voir [93]).

La solution de Balzer (voir [4]) possède les caractéristiques suivantes :
$\mathcal{T}(n) = 2n - 2$, $\mathcal{P}(\text{balzer}) = 165$, $|Q| = 8$, $W(n) = \mathcal{O}(n^2)$, récursive bidirectionnelle, symétrique double, infinité de signaux.

Les deux solutions correspondent à un même schéma, illustré par la figure 2.4, mais deux implémentations différentes. Il est à noter que les deux résultats ont été obtenus indépendamment.

L'idée est la suivante : prenons une solution de type Minsky-McCarthy, plutôt que de lancer deux solutions miroirs l'une de l'autre à l'emplacement de la coupure, pourquoi ne pas lancer dès le rebond du signal de pente 1 sur l'autre bord une solution miroir ? Si tel est le cas, une coupure en $3n/4$ sera

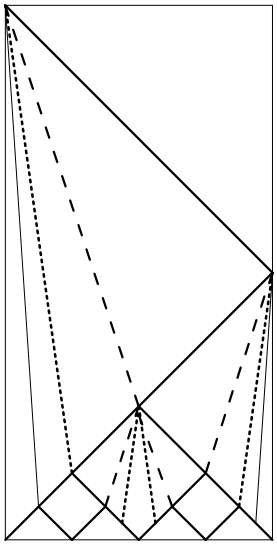


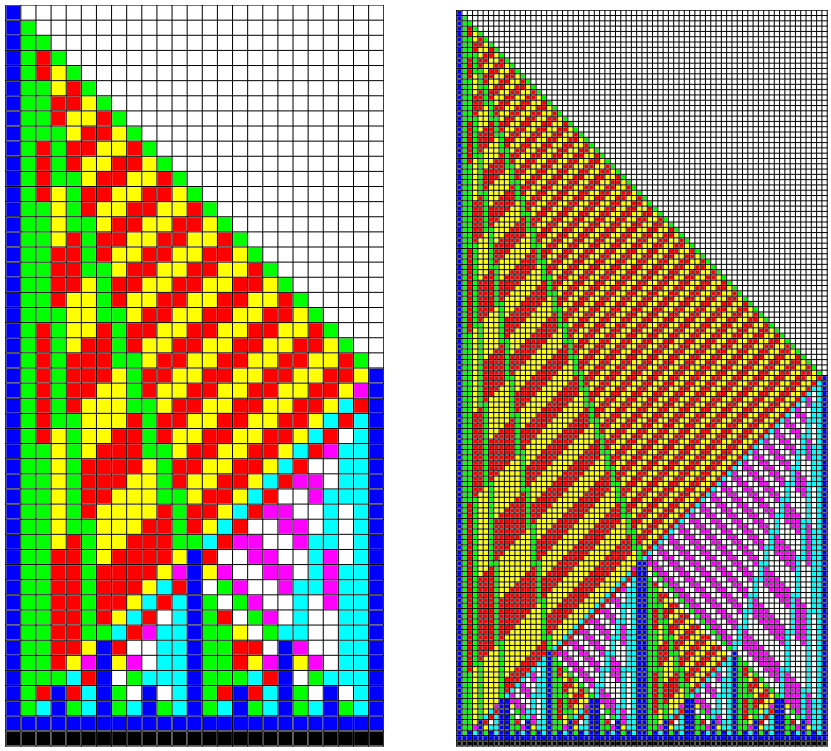
FIG. 2.4 – L'algorithme de Waksman/Balzer

obtenue (par l'opération miroir) au temps $n + 3n/4$, donc pour des raisons de symétrie il faudra construire dans le calcul d'origine une coupure en $n/4$ au temps $n + 3n/4$, et donc par symétrie une coupure en $3n/8$, etc. La limite est alors non plus de $3n$ mais $2n$.

La difficulté réside ensuite dans l'implémentation (voir la figure 2.5) car il faut désormais à partir du point d'origine générer une infinité potentielle de signaux de pente 1, 3, 7, $2^i - 1$, etc. L'astuce désormais connue est de remarquer que cette famille peut être obtenue par un mécanisme similaire à celui permettant d'obtenir par démultiplication et à partir d'une horloge donnée une sous-horloge. La figure A.1 de l'Annexe contient la table de transition de la solution.

2.1.6.3 Gerken

Hans-D. Gerken (voir [19]) a conçu deux solutions très différentes. Une première qui est une implémentation d'un schéma de type Waksman-Balzer. Une seconde dont l'algorithme s'apparente à celui de Goto. Malheureusement pour la seconde, le document la décrivant est difficile à obtenir. La figure 2.6 illustre le fonctionnement de la première solution sur une ligne de longueur 25.



(a) $l = 25$ (b) $l = 70$

FIG. 2.5 – La solution de Balzer

Cette solution a les caractéristiques suivantes :
$\mathcal{T}(n) = 2n - 2$, $\mathcal{P}(\text{gerken}) = 105$, $|Q| = 7$, $W(n) = \mathcal{O}(n^2)$, récursive bidirectionnelle, symétrique double, infinité de signaux.

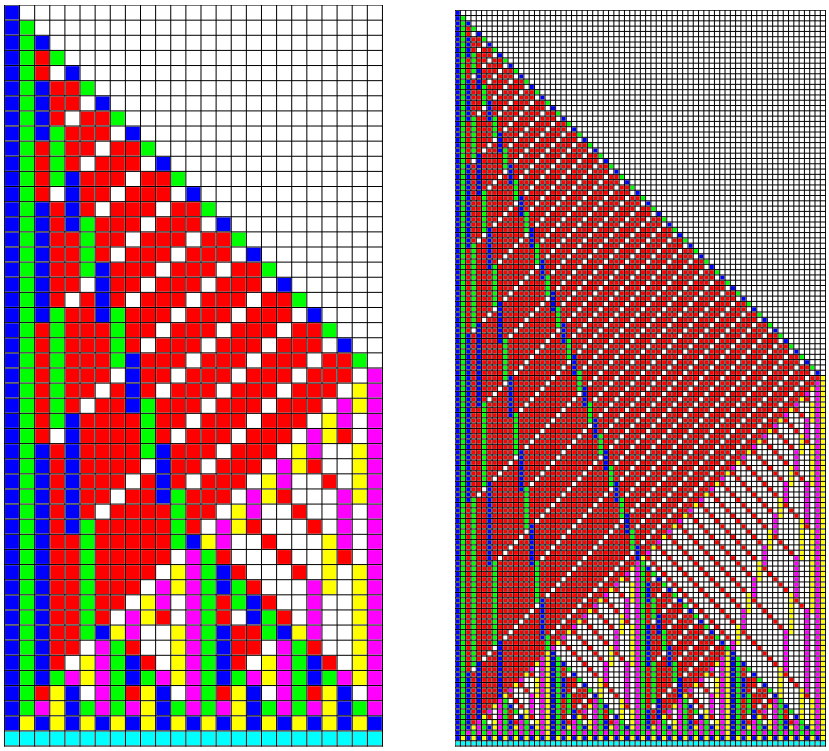
On trouvera en annexe la fonction de transition de la solution de Gerken.

2.1.6.4 Mazoyer

Cette solution a les caractéristiques suivantes :
$\mathcal{T}(n) = 2n - 2$, $\mathcal{P}(\text{mazoyer}) = 119$, $|Q| = 6$, $W(n) = \mathcal{O}(n^2)$, récursive unidirectionnelle, non symétrique, infinité de signaux.

Jacques Mazoyer est le concepteur de cette solution très astucieuse qui présente de nombreux intérêts (voir [51]). Outre le record et l'horizon sans doute indépassable qu'elle constitue ses caractéristiques sont particulières.

La construction est obtenue non pas par un algorithme de type « divide



(a) $l = 15$ (b) $l = 70$

FIG. 2.6 – La solution de Gerken

and conquer », mais par un emboîtement récursif très ingénieux ainsi que l'illustre la figure 2.7.

Les signaux, en nombre infini, ne sont plus représentés par des états particuliers mais par l'état quiescent lui-même : les signaux sont « en creux ». De plus, ils ne sont pas uniformes, *i.e.* bien que représentant des pentes rationnelles la forme obtenue ne représente pas ce à quoi l'on pourrait s'attendre et qui correspondrait à une discrétisation habituelle du signal continu correspondant.

De plus, Jacques Mazoyer a formellement prouvé la correction de sa solution, et c'est à notre connaissance la première preuve mathématique d'une solution au problème du Firing Squad. Les lecteurs de cette preuve n'ont pu que remarquer la grande difficulté à la saisir dans son ensemble ; et l'impression que malheureusement elle ne dit rien sur le fonctionnement de la

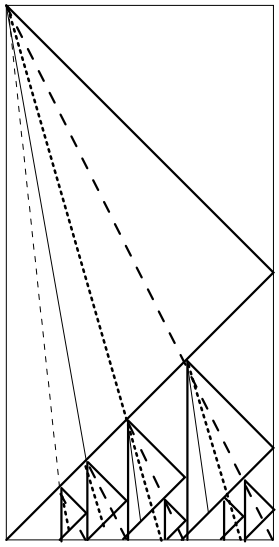


FIG. 2.7 – L'algorithme de Mazoyer

solution (en tous cas pas aussi bien que l'observation du fonctionnement). Il est à noter que cette preuve a elle-même été vérifiée par Jean Duprat (voir [15]) à l'aide d'un système de vérification de preuve, Coq.

On trouvera en annexe la fonction de transition de la solution de Mazoyer.

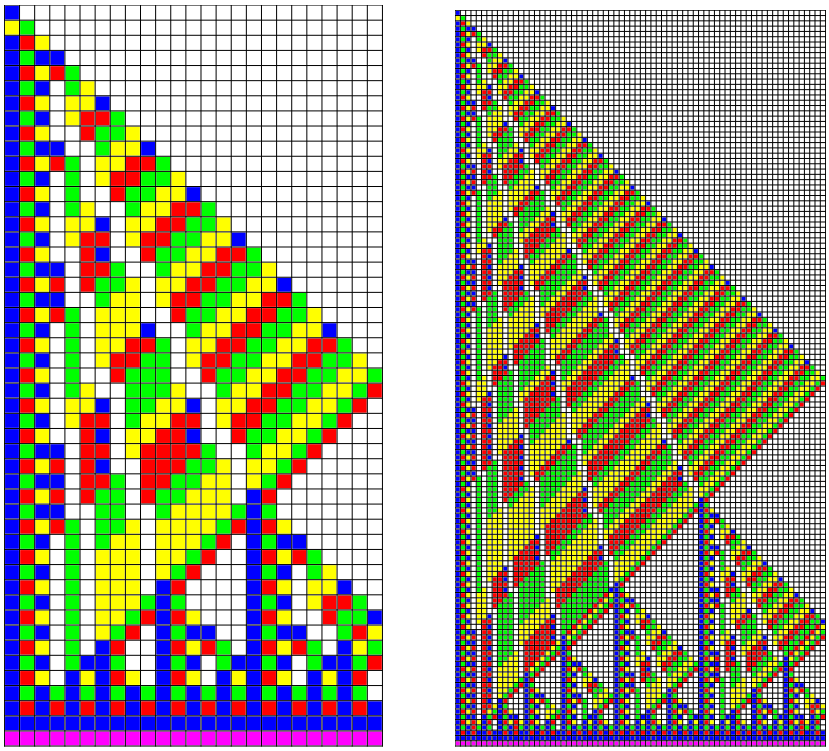
2.1.6.5 Noguchi

Cette solution a pour caractéristiques :
$\mathcal{T}(n) = 2n - 2$, $\mathcal{P}(\text{noguchi}) = 123$, $|Q| = 8$, $W(n) = \mathcal{O}(n^2)$, récursive bidirectionnelle, symétrique double, infinité de signaux.

La solution de Kenishiro Noguchi (voir [63]) n'apporte rien de nouveau en ce qui concerne l'algorithmique employée, il s'agit d'un schéma de type Waksmann/Balzer, mais elle peuple le zoo des implémentations en peu d'états puisqu'elle n'en utilise que 8 (autant que Balzer). L'obtention de la famille d'horloges permettant d'obtenir l'infinité de signaux est assez remarquablement simple.

Son intérêt réside dans l'existence d'une preuve formelle de son fonctionnement : la seconde preuve de ce type.

La figure 2.9 fournit un exemple d'exécution pour des lignes 25 et 70. On trouvera en annexe la fonction de transition de cette solution.



(a) $l = 25$ (b) $l = 70$

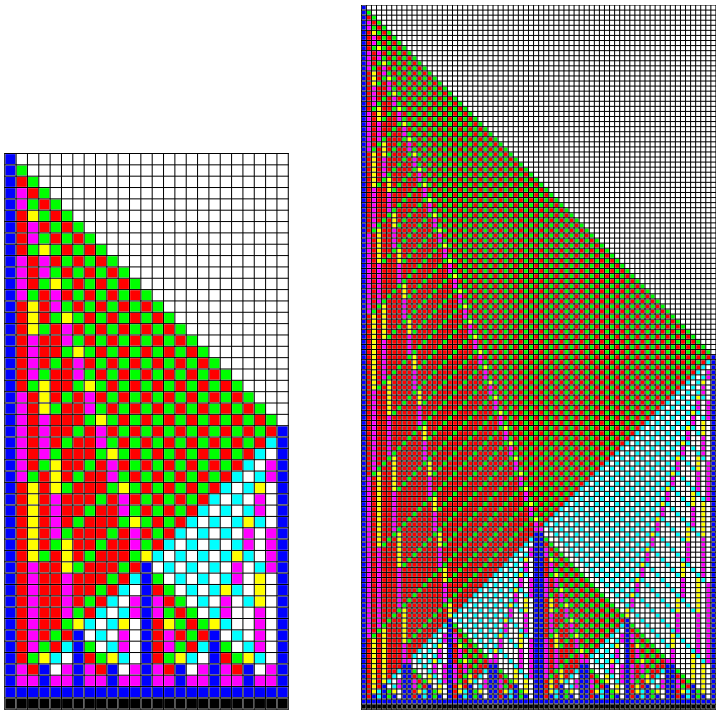
FIG. 2.8 – La solution de Mazoyer

2.1.7 Les avancées récentes sur les solutions en peu d'états

Depuis la solution inspirée des idées de Laurent Pierre (voir [65]), de nouvelles solutions ont été recherchées en « oubliant » le temps minimal. Ainsi divers auteurs ont contribué à peupler le zoo des solutions en peu d'états. Settle (voir [81, 80, 79]), Umeo (voir [85, 90, 91, 92, 94, 95]) et Yunès (voir [108, 113, 114]) ont construit des solutions avec peu d'états synchronisantes en temps non-minimal.

2.1.7.1 Yunès

Depuis Minsky-McCarthy et la chasse aux solutions minimales en temps et nombre d'états, aucune solution non minimale en temps n'avait été étudiée.



(a) $l = 25$ (b) $l = 70$

FIG. 2.9 – La solution de Noguchi

En 1993, j'ai conçu une solution de synchronisation de type Minsky (voir [108] et la figure 2.10) dont les caractéristiques sont les suivantes :
$\mathcal{T}(n) = 3n + \mathcal{O}(log(n))$, $\mathcal{P}$(yunes) $= 135$, $|Q| = 7$, $W(n) = \mathcal{O}(n.log(n))$, récursive bidirectionnelle, non symétrique, finitude de signaux.

La difficulté réside dans le codage utilisé pour les signaux qui conduit à l'obtention de signaux d'épaisseur 2. Cette solution a sans doute relancé l'intérêt de la quête de solutions avec peu d'états et non nécessairement en temps-minimal. L'article en question contient les détails du calcul de la complexité en temps de l'algorithme.

On trouvera en annexe les tables de transition de cette solution.

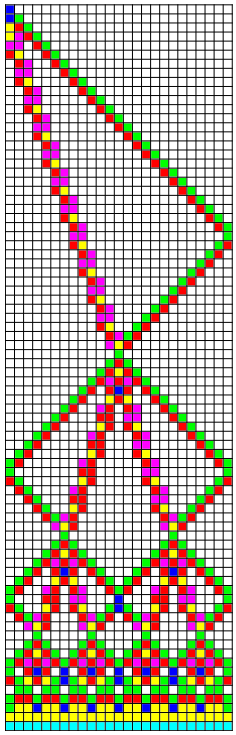


FIG. 2.10 – La solution de Yunès : une exécution pour la longueur 25

2.1.7.2 Settle & Simon

La solution de Amber Settle et Janos Simon (voir [81]) est intéressante. Bien que l'on puisse faire entrer cette solution dans notre classification, il faut savoir que celle-ci ne rend pas compte de sa véritable nature, néanmoins : $\mathcal{T}(n) = 3n - 2$, $\mathcal{P}(\text{settle}) = 127$, $|Q| = 6$, $W(n) = \mathcal{O}(n^2)$, ne peut être vraiment qualifiée en tant récursive ou itérative, non symétrique, infinité de signaux.

L'astuce utilisée est d'employer des transitions non définies de la solution de Mazoyer afin d'obtenir un signal de pente 1 traversant d'un côté à l'autre la ligne ; puis lorsque le signal touche l'autre bord de lancer la solution originelle.

Bien que constituant un réel progrès, une certaine insatisfaction apparaît car il ne s'agit pas d'un algorithme qui converge en $3n$ mais d'une séquence de deux algorithmes enchaînés l'un après l'autre. Toutefois, à partir de cette solution j'ai pu me demander si certaines solutions existantes pouvaient être étendues de la sorte (voir plus loin).

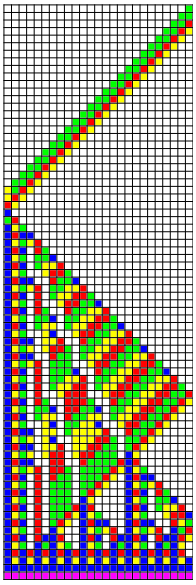


FIG. 2.11 – La solution de Settle/Simon : une exécution de longueur 25

On trouvera en annexe la fonction de transition de la solution.

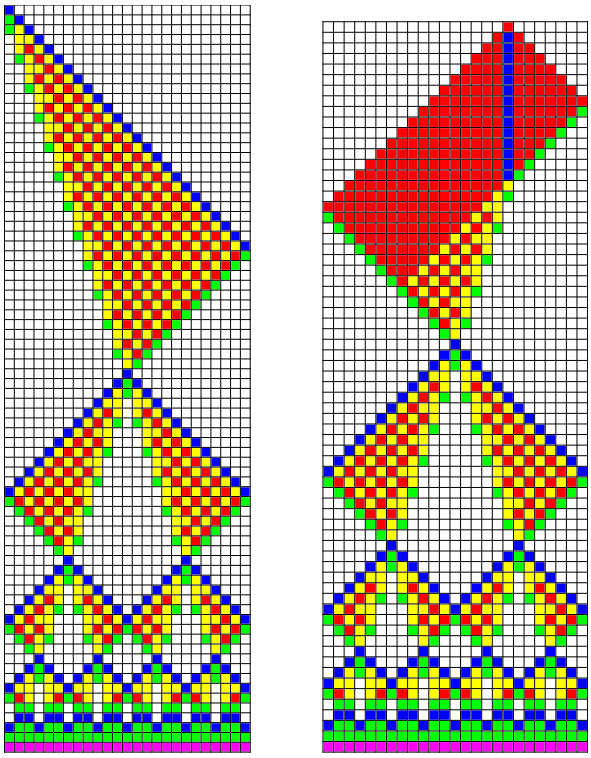
2.1.7.3 Umeo

Ses caractéristiques sont : $\mathcal{T}(n) = 3n+\mathcal{O}(log(n))$, $\mathcal{P}(\text{umeo}) = 78$, $|Q| = 6$, $W(n) = \mathcal{O}(n^2)$, récursive bidirectionnelle, symétrique, finitude de signaux.

La solution de Hiroshi Umeo (voir [92]) est intéressante à plus d'un titre. Tout d'abord elle est l'implémentation d'un schéma de Minksy-McCarthy en seulement 6 états. Ensuite, on peut remarquer quelle est symétrique, et qu'elle permet donc de résoudre le problème non restreint (départ à gauche ou à droite sur la ligne) ainsi que sur l'anneau.

Mais son autre particularité est qu'il est possible de l'étendre afin qu'elle synchronise quelle que soit la position du général. Pour obtenir ce résultat Umeo a étendu la fonction de transition originale de sa solution de façon que le mécanisme de reconstruction du « divide-and-conquer » puisse prendre place (il a toutefois été nécessaire de modifier l'une des transitions de la fonction originale).

Pour cette extension les caractéristiques sont : $\mathcal{T}(n) = 3n-p+\mathcal{O}(log(n))$ (où p est la position initiale du général), $\mathcal{P}(\text{umeo2}) = 115$, $|Q| = 6$, $W(n) = \mathcal{O}(n^2)$, récursive bidirectionnelle, symétrique, finitude de signaux.



(a) $l = 25$ (b) $l = 25$, $p = 18$

FIG. 2.12 – La solution de Umeo

Encore une fois, c'est en m'inspirant d'une telle idée que les résultats de robustesse ont pu être établis (voir § 2.1.8.2).

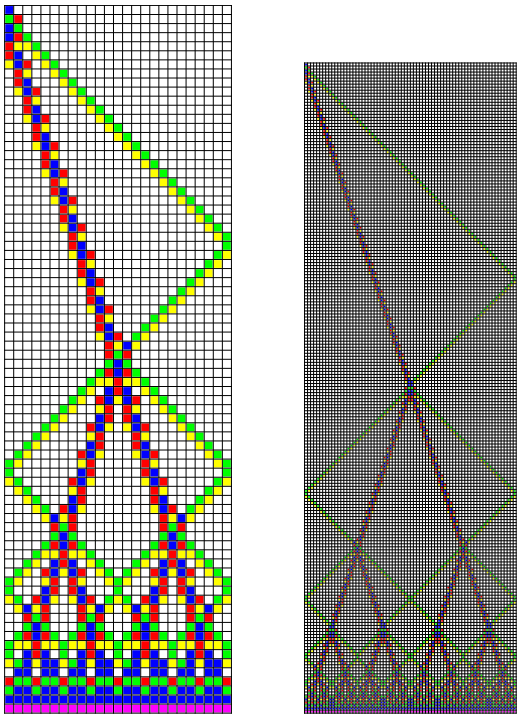
2.1.7.4 Yunès

Ses caractéristiques sont : $\mathcal{T}(n) = 3(n-1) + \lceil log(n) \rceil$, $\mathcal{P}(\text{yunes6}) = 132$, $|Q| = 6$, $W(n) = \mathcal{O}(n.log(n))$, récursive bidirectionnelle, symétrique, finitude de signaux.

Cette solution, voir la figure 2.13, a été obtenue indépendamment de celle de Umeo, nos travaux parallèles nous ont conduit lui à construire une solution qui travaille en n^2 et moi une solution qui travaille en $n.log(n)$. Le fait remarquable est que nous ayons chacun construit une solution en 6 états et symétrique. La brisure de symétrie utilisée par Mazoyer a un temps été

considérée comme nécessaire pour économiser des états, ce qui est peut-être nécessaire pour une solution en temps minimal, mais qui est clairement faux pour les autres. Le temps de synchronisation est de $3n + \mathcal{O}(log(n))$.

Cette solution est formellement prouvée (voir [114]), et la preuve est singulièrement simple. Peut-être est-ce dû à la simplicité de l'algorithme ?



(a) $l = 25$ (b) $l = 70$

FIG. 2.13 – La solution de Yunès

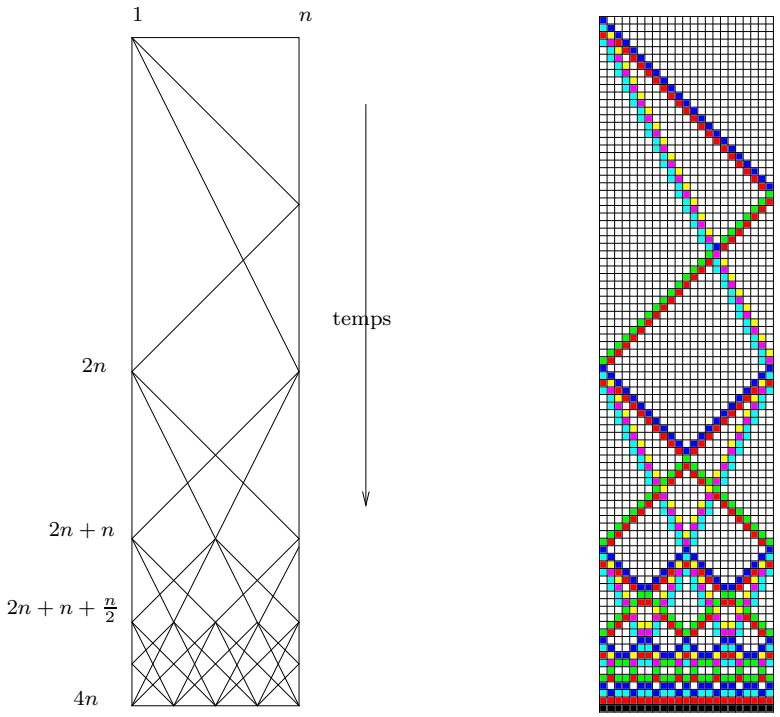
On trouvera en annexe la figure A.8 qui contient la fonction de transition de cette solution.

2.1.7.5 Plus loin... la synchronisation linéaire

En ce point, on peut se demander jusqu'où, en temps, il est possible d'aller avec peu d'états ; si possible en utilisant des algorithmes qui convergent intrinsèquement vers le temps à considérer (sans astuce à la Settle-Simon). Mes récentes découvertes sont surprenantes, à la fois de simplicité et d'élégance

puisque les algorithmes (que l'on peut unifier comme on le verra) sont tous inspirés de la coupure de Minsky-McCarthy.

On trouvera en figure 2.14 la solution aux caractéristiques suivantes : $\mathcal{T}(n) = 4n + \mathcal{O}(log(n))$, $\mathcal{P}$(yunes4n) = 195, $|Q| = 8$, $W(n) = \mathcal{O}(n.log(n))$, récursive bidirectionnelle, symétrique, finitude de signaux. La figure A.9 de l'annexe contient la fonction de transition.

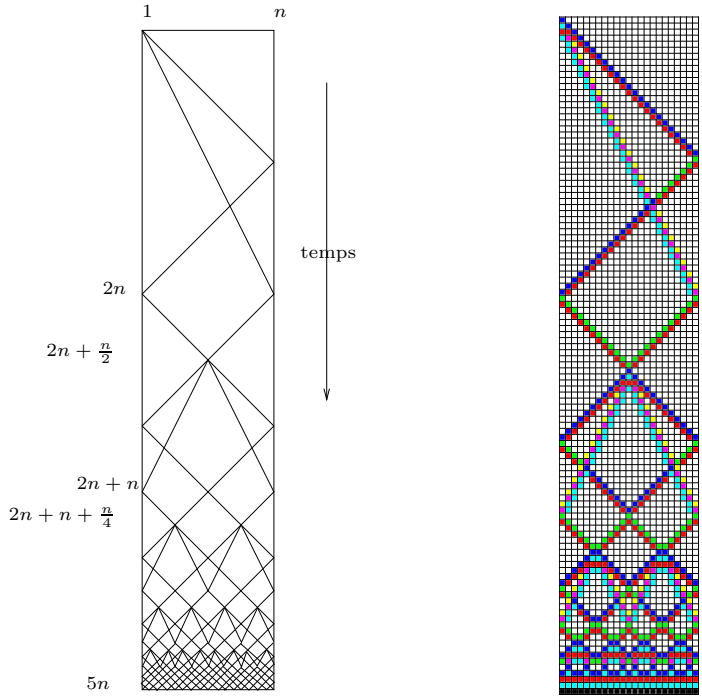


(a) schéma (b) $l = 23$

FIG. 2.14 – Une solution à 8 états en temps $4n$

La figure 2.15 contient une description de la solution aux caractéristiques suivantes : $\mathcal{T}(n) = 5n + \mathcal{O}(log(n))$, $\mathcal{P}$(yunes5n) = 177, $|Q| = 8$, $W(n) = \mathcal{O}(n.log(n))$, récursive bidirectionnelle, symétrique, finitude de signaux. La figure A.10 de l'annexe contient la fonction de transition.

On trouvera en figure 2.16(b) une solution aux caractéristiques suivantes : $\mathcal{T}(n) = 4n+\mathcal{O}(log(n))$, $\mathcal{P}$(yunes4nbis) = 161, $|Q| = 7$, $W(n) = \mathcal{O}(n.log(n))$, récursive bidirectionnelle, symétrique, finitude de signaux. La figure A.11 de l'annexe contient la fonction de transition.



(a) schéma (b) $l = 23$

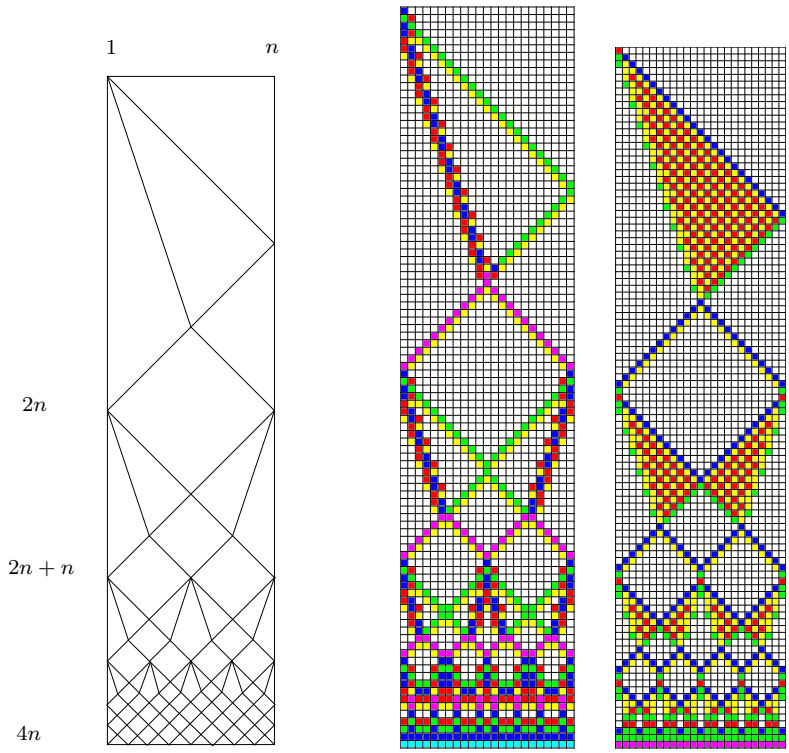
FIG. 2.15 – Une solution à 8 états en $5n$

La figure 2.16(c) est une implémentation aux caractéristiques suivantes : $\mathcal{T}(n) = 4n + \mathcal{O}(log(n))$, $\mathcal{P}$(yunes4n = 95, $|Q| = 6$, $W(n) = \mathcal{O}(n^2)$, récursive bidirectionnelle, symétrique, finitude de signaux. La figure A.12 de l'annexe contient la fonction de transition.

On trouvera en figure 2.17 une solution aux caractéristiques suivantes : $\mathcal{T}(n) = 6n + \mathcal{O}(log(n))$, $\mathcal{P}$(yunes6n = 136, $|Q| = 6$, $W(n) = \mathcal{O}(n.log(n))$, récursive bidirectionnelle, symétrique, finitude de signaux. La figure A.13 de l'annexe contient la fonction de transition.

2.1.7.6 Encore plus loin... la synchronisation polynomiale ?

Nous envisageons d'étudier la possibilité de synchroniser en temps polynomial avec peu d'états. Bien entendu il serait intéressant de le faire en s'aidant d'un algorithme qui converge intrinsèquement en temps polynomial. Car nous savons depuis les résultats d'Olivier Heen (voir [31]) que tout po-



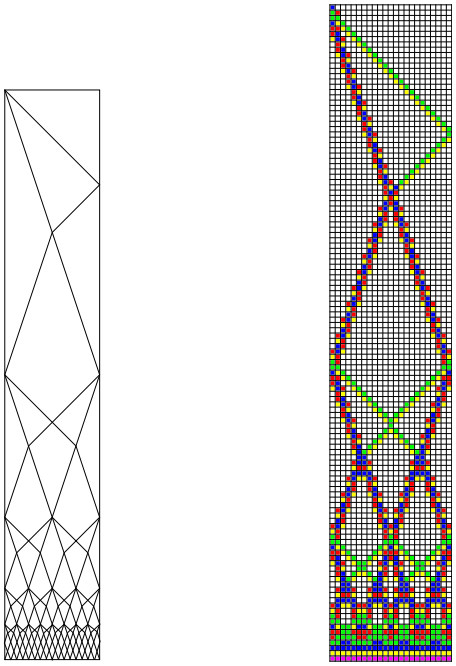
(a) schéma (b) $l = 23$ (c) $l = 25$

FIG. 2.16 – Une solution alternative en temps $4n$

lynôme $P(n)$ tel que $\forall\, n \geq 1$, $P(n) \geq 2n - 2$ est le temps de synchronisation d'un certain automate cellulaire, même lorsque les cœfficients du polynôme sont dans $\mathbb{Z}$ et non dans $\mathbb{N}$.

2.1.8 Les extensions

Il existe de très nombreuses variantes ou extensions du problème. La première est sans doute la synchronisation dans des espaces de dimensions supérieures, parmi lesquelles la synchronisation de grilles, de rectangles, de cubes et autres parallélépipèdes. Ensuite l'étude de la synchronisation sur des graphes particuliers ou quelconques.



(a) schéma (b) $l = 23$

FIG. 2.17 – Une solution alternative en temps $6n$ et à 6 états

2.1.8.1 La tolérance aux pannes

On retrouvera cette problématique dans nombre d'articles séminaux d'informatique. John von Neumann (voir [102]), Edward Moore et Claude Shannon (voir [58]) ou plus récemment Nicholas Pippenger (voir [66]) se sont intéressés au problème de la tolérance aux pannes. Dans de nombreux cadres généraux, la problématique est sans espoir, car il n'existe aucun système qui puisse continuer sa route normalement quel que soit le nombre d'erreurs ou le type d'erreur. Toutefois, mais sous certaines hypothèses, il est possible d'envisager d'en corriger certaines. Ainsi, aujourd'hui de nombreux dispositifs sont présents dans les ordinateurs de façon qu'ils puissent fonctionner normalement (fournir un calcul répondant aux spécifications contenues dans le programme) alors même que de fréquentes erreurs peuvent s'y produire.

Dans ce type de problèmes on distingue : les fautes, les erreurs, et les défaillances. C'est-à-dire la cause, l'incohérence qui s'en suit elle même susceptible de conduire à la défaillance du système. Les défaillances sont de plu-

sieurs ordres. La panne franche est la plus « visible », le système ne répond plus (une coupure d'électricité provoque généralement la panne franche d'un appareil électrique). La panne par omission est ennuyeuse car le système « oublie » de répondre à certaines sollicitations (l'utilisateur s'énerve assez vite, ne sachant pas si cela vient de lui ou de la machine). Pour la panne dite byzantine, dans ce cas le système fait n'importe quoi ; celle-ci est particulièrement désagréable et perverse car le système peut même faire quelque chose de vraisemblable. Bien entendu ces diverses pannes peuvent avoir un caractère transitoire, intermittent ou permanent.

Dans les cas qui nous intéressent, le modèle de faute que nous considérons est particulier. Nous supposons que les cellules possèdent un mécanisme de détection de panne et que lorsque celui-ci la détecte, la cellule se met définitivement dans un mode très particulier de « neutralité » : la cellule se contente de passer les informations d'un côté à l'autre sans exécuter un quelconque calcul. La difficulté de conception d'un algorithme fonctionnant sur ce type de modèle réside dans le fait que la géométrie du calcul se trouve bouleversée à la rencontre d'une telle région, car les signaux sont localement transmis à la « vitesse de la lumière[7] ». La neutralité est donc relative à la fonction de transition pas en ce qui concerne les effets produits !

On notera l'existence du résultat de Martin Kutrib et Roland Vollmar (voir [46, 47]) qui indique l'impossibilité d'obtenir une solution au firing squad dans un environnement défectueux sans hypothèse de restriction.

Les solutions que nous exposons ici sont en temps linéaire, mais il existe d'autres résultats statuant sur la complexité en temps sans conditions particulières (voir [28, 39, 42, 73, 98], tous au-delà du temps linéaire).

Umeo

Cet algorithme (voir [86]) permet d'obtenir une synchronisation en temps minimal quand bien même certaines cellules sont défaillantes. La faisabilité réside dans la restriction employée, ici la répartition des cellules défectueuses est telle que si l'on considère la ligne comme une alternance de segments de cellules soient toutes fonctionnelles ou bien toutes défectueuses. Alors pour une alternance de cellules fonctionnelles puis défectueuses le nombre de cellules fonctionnelles est plus grand que celui de cellules défectueuses.

[7] On appelle « vitesse de la lumière » dans les automates cellulaire la vitesse la plus rapide à laquelle l'information peut-être transmise. Dans le firing squad considéré ici, il s'agit d'un signal de pente 1.

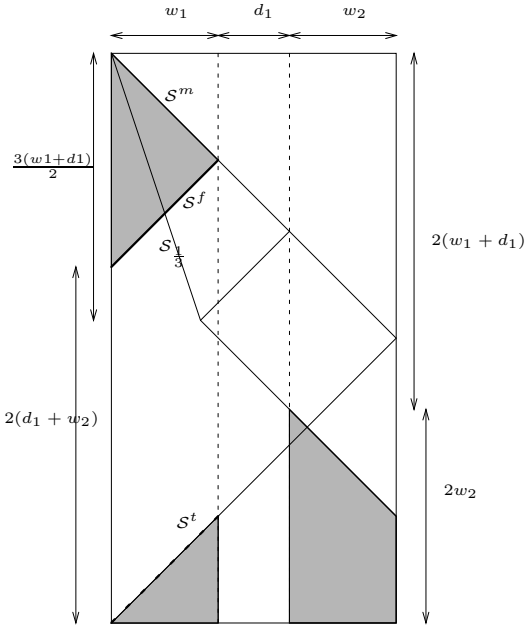


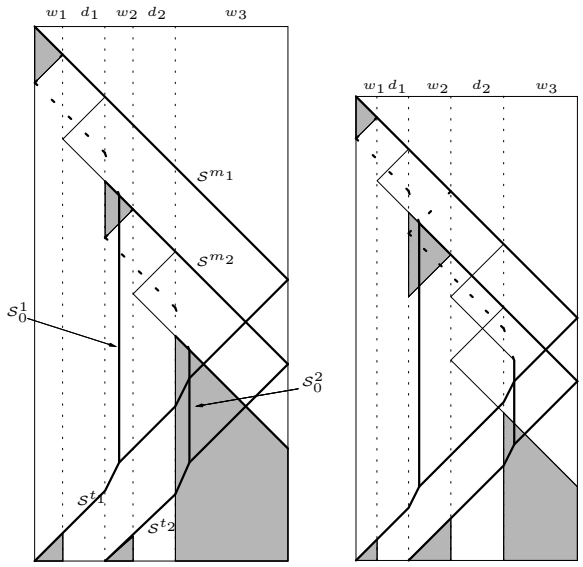
FIG. 2.18 – Solution tolérante aux pannes d'Umeo

De plus la construction est obtenue en utilisant un mécanisme de gel/dégel (voir [73]) du calcul en cas de nécessité. Chaque segment fonctionnel calcule sa propre synchronisation, laquelle est découpée en deux parties par le mécanisme de gel/dégel, de sorte que l'ensemble des parties dégelées se synchronisent ensemble. La durée du gel est fonction d'un retard calculé à l'aide des distorsions induites par les traversées des segments défectueux comme on peut l'observer dans la figure 2.18.

Yunès

Cet algorithme (voir [110]) permet de s'affranchir en grande partie de la contrainte d'alternance de la solution d'Umeo. Dans notre cas, la répartition des cellules défectueuses est plus globale, toutefois, il doit rester dans une alternance donnée un espace suffisant pour construire quelques calculs élémentaires (une différence de longueur). Le temps global de synchronisation n'est plus minimal mais dépend de ces conditions très défavorables où il y a localement plus de cellules défectueuses que fonctionnelles.

Un algorithme mélangeant les deux possibilités offertes par Umeo et



(a) Non optimisé (b) Optimisé

FIG. 2.19 – Solution tolérante aux pannes de Yunès

Yunès a été conçu, se reporter à la figure 2.20.

2.1.8.2 La robustesse

Ici les résultats que j'ai pu obtenir sont sans précédent, bien que les idées puissent être retrouvées disséminées ci et là dans diverses solutions conçues ces dernières années. Ce sont notamment les solutions de Settle et Umeo qui m'ont mis la puce à l'oreille. Elles correspondent toutes les deux a une extension de la fonction de transition d'une solution originelle. Settle a étendu une solution de Mazoyer pour obtenir une solution en $3n$ où le général est à droite (voir figure 2.11) et Umeo car il a étendu la fonction de transition d'une de ses solutions pour obtenir une solution où le général peut être placé n'importe où sur la ligne. Le fait remarquable est que ces deux extensions sont conservatives, elles ne modifient pas la fonction originelle. D'autre part, le général employé pour obtenir la nouvelle fonctionnalité est différent du général à employer pour exécuter la fonction originelle. J'ai, sur ces deux idées, conçu des solutions étendues pour toutes les solutions existantes connues.

Il a d'abord été possible de construire (voir [111, 112]) des solutions per-

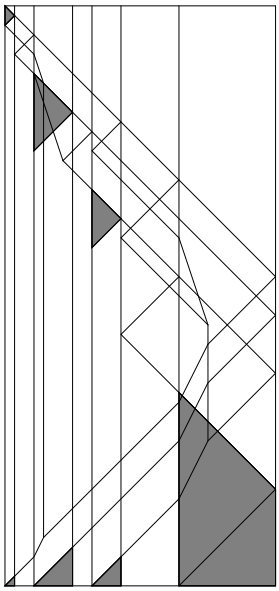


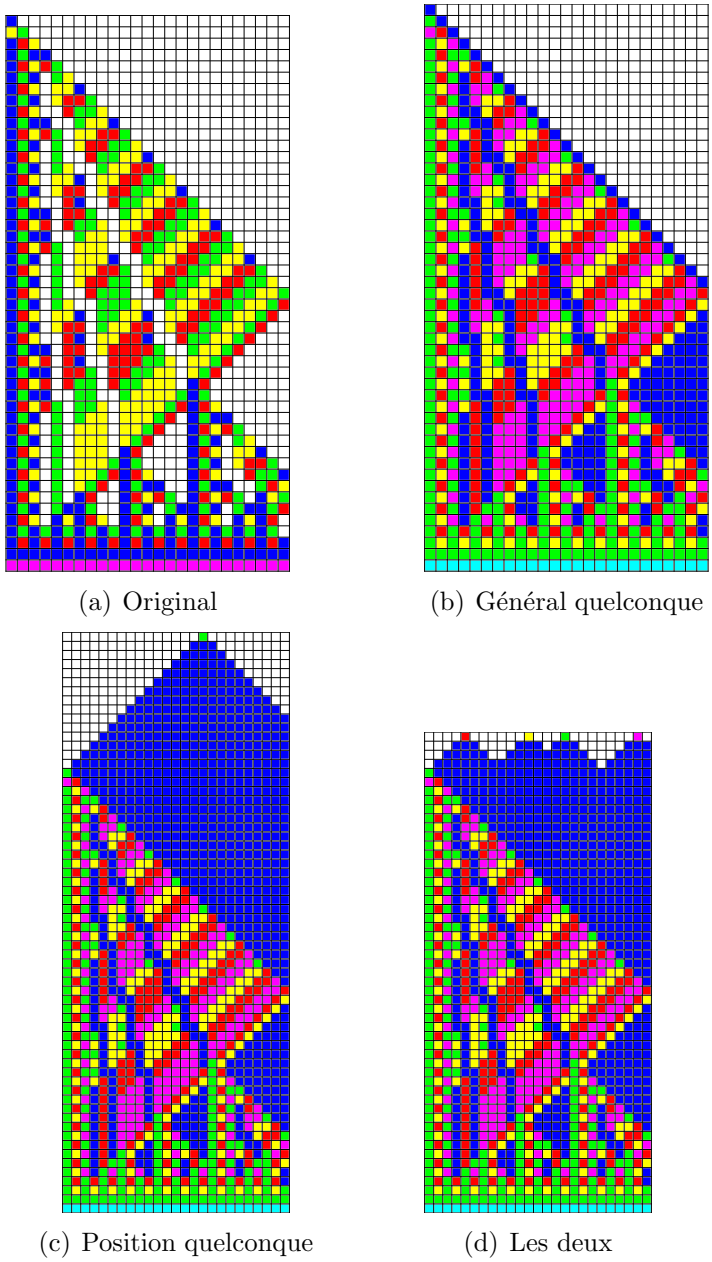
FIG. 2.20 – Solution tolérante aux pannes d'Umeo/Yunès

mettant d'obtenir la synchronisation à partir de n'importe quel état comme état du général (sauf l'état quiescent et l'état feu), et ce pour les solutions de Mazoyer, Gerken, Balzer, Noguchi, Yunès, Umeo, Settle.

Ensuite, et pour chacune des mêmes solutions originelles, l'extension a consisté à permettre de démarrer la synchronisation quelle que soit la position du général.

Dans les deux cas, il n'a pas été possible de trouver de mécanisme général (voir [115]) permettant d'obtenir ces extensions, mais le fait remarquable est que cela a néanmoins pu être réalisé dans les cas particuliers mentionnés alors qu'a priori ces solutions n'ont pas été conçues de la sorte. On notera aussi qu'il s'est révélé impossible d'obtenir les deux extensions à la fois. Ce résultat négatif devient positif et effectif lorsqu'on s'autorise à ajouter un état supplémentaire (voir [115]) comme l'illustre la figure 2.21.

Mieux, l'extension proposée permet accessoirement de résoudre le problème à plusieurs généraux (voir [77, 78]) y compris décalés dans le temps (ce problème est dénommé A-MG-FSSP, pour Asynchronous Multi General FSSP. L'asynchronisme faisant référence aux départs décalés dans le temps des généraux).



(a) Original (b) Général quelconque

(c) Position quelconque (d) Les deux

FIG. 2.21 – Extensions génériques

2.1.9 Vers une réécriture du problème

Les différentes solutions que nous avons rencontrés suggèrent une possible réécriture du problème (ou une simple relecture ?).

2.1.9.1 Surfusion et métastabilité

En effet, le problème original suppose que le général soit un état dédié particulier (c'est en tous cas une interprétation constante) et positionné à gauche (ou à droite peut importe).

Une solution où le général est situé n'importe où sur la ligne est considérée comme une solution à une extension du problème, or les différents résultats obtenus montrent que l'on devrait considérer une telle situation comme ordinaire puisque toute solution peut-être étendue en ce sens et à faible coût (voir [112, 115]).

De plus, on remarquera qu'il semble possible pour toutes les solutions ordinairement construites de les étendre de sorte que la synchronisation s'établisse quelque soit l'état utilisé pour le général, *i.e.* que pour une implémentation donnée il soit possible d'obtenir la synchronisation quelque soit l'état de départ.

Cette dernière propriété peut paraître étrange, mais elle semble correspondre à une propriété similaire de certains phénomènes d'émergence du monde physique. Que l'on songe par exemple à la surfusion qui définit l'état d'une matière demeurant en phase liquide alors même que sa température interne est en dessous du point de solidification (l'eau peut être surfondu jusqu'à -39°C !).

Il est connu que cet état stable est précaire, on dit métastable, et qu'une légère perturbation (de quelque nature qu'elle soit) peut permettre le passage instantané à l'état stable correspondant (solidification dans notre exemple). Nous connaissons l'exemple ordinaire des brouillards givrants. En chimie on connaît l'exemple des structures métastables obtenues puis conservées par des procédés comme la trempe des aciers.

Dans notre cas, un raccourci intellectuel peut nous conduire à interpréter la propriété comme suit : la ligne est dans un état métastable (entièrement quiescente), une énergie d'activation est fournie au système (une cellule est positionnée dans un état quelconque) et le système converge vers un autre état stable (celui de synchronisation).

2.1.9.2 Pas toutes les lignes

On peut aussi être amené à ne plus considérer la synchronisation de toutes les longueurs possibles, mais seulement une infinité. En ce cas, des résultats non publiés récents montrent que cette voie est aussi très prometteuse. Umeo (voir [94]) a construit une solution en seulement 5 états, permettant de synchroniser toutes les lignes de longueur 2^n (Mazoyer l'avait aussi mais n'a jamais réellement communiqué à ce propos). Et de façon indépendante, Umeo et Yunès ont construit, chacun indépendamment, une solution à 4 états (différente dans les deux cas) permettant pour l'un de synchroniser des lignes de longueur 2^n et pour l'autre celles de longueur $2^n + 1$ (les publications sont en cours de rédaction [117]), et pour s'en faire une idée le lecteur est renvoyé à l'observation de la figure 2.22.

On peut envisager de définir une complexité de Kolmogorov pour les solutions ne fonctionnant que sur une infinité de lignes mais pas toutes. Par exemple, la complexité de Kolmogorov du firing squad sur les lignes de longueur 2^n est $\mathcal{K}_{2^{\mathbb{N}}}(FSSP) \leq 32$.

Le travail de cette solution est en $3n^{log(3)}$, un exemple de valeur fractale dont nous avons déjà parlé. En effet la propriété du travail de l'automate cellulaire sous-jacent (qui calcule le triangle de Pascal modulo 2) est $W_{\text{Pascal}}(2^{n+1}) = 3.W_{\text{Pascal}}(2^n)$ donc que $W_{\text{Pascal}}(2^n) = 3^n$, le travail total de la synchronisation est donc $W(2^n) = 3^{n+1}$ et donc $W(m) = 3.m^{log(3)}$ pour $m \in \{2^n,\, n \in \mathbb{N}\}$.

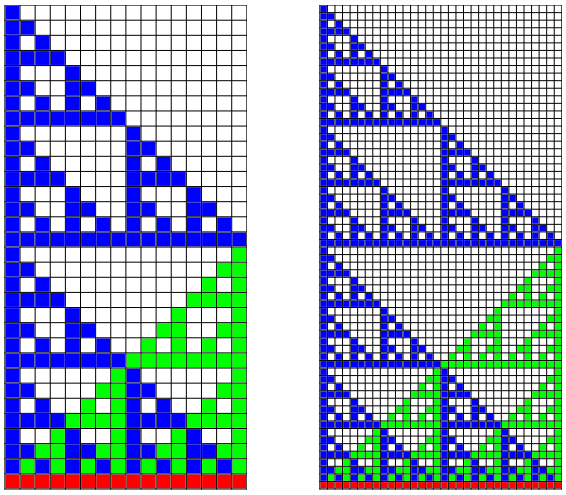
2.1.9.3 La réversibilité, La conservation

Katsunobu Imai et Kenichi Morita ont construit deux solutions du firing squad aux propriétés intéressantes.

Tout d'abord (voir [37]) il est possible de construire une solution au firing squad qui soit réversible[8]. Évidemment, il faut pour cela modifier l'énoncé du problème de sorte que « l'état de feu » ne soit plus un unique état. La synchronisation est, dans leur cas, obtenue par une configuration où toutes les cellules sont dans un état qui ne soit jamais apparu avant (la subtilité réside dans le fait qu'il ne soit pas nécessaire que toutes soient dans le même état !).

Ensuite (voir [38]) il est possible de construire une solution du firing

[8]La réversibilité est une préoccupation « physique », elle est liée à la problématique de la conservation de l'énergie, d'où son intérêt.



(a) $l = 16$ (b) $l = 32$

FIG. 2.22 – Une solution à 4 états

squad qui conserve une quantité globale (ces solutions sont dites « number-conservative »). Les états codent des nombres, et la somme globale de ces nombres est conservée tout au long du calcul.

Nous pensons que ces entorses à la définition du problème n'en sont pas, et que sans doute elles pourraient, au moins en ce qui concerne la réversibilité, faire partie de la définition du problème.

2.1.9.4 Un problème réécrit

Nous proposons donc que le problème soit réécrit de la façon suivante :

Construire un automate cellulaire

$$\mathcal{A} = (Q, \delta) \text{ avec } Q = \{q, e_1, \ldots, e_k\}$$

de sorte que pour une infinité de lignes (la longueur étant notée n) :

- si la configuration de départ est de la forme $\overbrace{q \ldots q}^{p-1} x \overbrace{q \ldots q}^{n-p}$, avec $x \in Q$, et $p \in [1, n]$,
- l'automate calcule au temps $\mathcal{T}(n)$ une configuration de la forme $c_1 c_2 \cdots c_n$ où $\forall i \in [1, n]$, $c_{i-1} c_i c_{i+1} \notin Dom(\delta)$. Le « feu » est

donc remplacé par une configuration quelconque mais « bloquante ».

2.2 Récapitulatif

Le tableau 2.1 résume la situation actuelle en ce qui concerne le firing squad sur la ligne finie.

2.3 Quoi faire encore ?

Parmi les possibilités envisageables nous songeons à étudier la robustesse à des changements de voisinage. Hidenosuke Nishio (voir [60, 61]) et Thomas Worsch (voir [106]) ont entamé ce type de travail dans des cadres plus généraux. Nous nous demandons simplement s'il existe des fonctions de synchronisation qui fonctionnent sur plusieurs voisinages.

La réversibilité telle qu'étudiée par Kenichi Morita, nous fait nous demander s'il n'existerait pas un schéma générique de firing squad réversible. On notera que le firing squad réversible obtenu par Imai et Morita (voir [37]) a inspiré la rédéfinition précédente du problème et notamment le remplacement de l'état de feu par une configuration de blocage.

Une définition du firing squad infini. La première difficulté est de définir correctement le problème. Nous pensons que la première bonne définition correspond à la synchronisation partant d'une ligne infinie périodique (les généraux sont espacés à intervalles réguliers). Des travaux sont en cours en ce qui concerne une possible a-périodicité de la configuration initiale.

2.4 Un logiciel

Dans le cadre de ces travaux, nous avons été amené à écrire un programme de manipulation des automates cellulaires linéaires finis qui nous a permis d'expérimenter et de créer les solutions ici décrites. L'ensemble des diagrammes espace-temps de ce mémoire a été généré à l'aide de ce logiciel. Ce programme est disponible à qui le souhaite, il suffit pour l'obtenir de le télécharger (voir [120]). Il n'est pour l'instant disponible que pour système `MacOSX`®. Les sources sont à réclamer directement auprès de l'auteur.

Solution	**algorithme**	$\mathcal{T}(n)$	$\mathcal{P}(n)$	$\lvert Q\rvert$	$\mathcal{W}(n)$	**type**	**symétrie**	**# signaux**
Goto	Goto	$2n-2$	$\leq 10^9$	$\leq 10^6$	$\mathcal{O}(n.log(n))$	Itérative	non	fini
Waksman	Wakman/Balzer	$2n-2$	202	16	$\mathcal{O}(n^2)$	Réc. Bidir.	double	infini
Balzer	Wakman/Balzer	$2n-2$	165	8	$\mathcal{O}(n^2)$	Réc. Bidir.	double	infini
Gerken	Wakman/Balzer	$2n-2$	105	7	$\mathcal{O}(n^2)$	Réc. Bidir.	double	infini
Gerken II	Goto	$2n-2$	?	?	$\mathcal{O}(n^2)$	Itérative	non	?
Mazoyer	Mazoyer	$2n-2$	119	6	$\mathcal{O}(n^2)$	Réc. Unidir.	non	infini
Noguchi	Waksman/Balzer	$2n-2$	123	8	$\mathcal{O}(n^2)$	Réc. Bidir.	double	infini
Yunès	Minsky/McCarthy	$3n+\mathcal{O}(log(n))$	135	7	$\mathcal{O}(n.log(n))$	Réc. Bidir.	non	fini
Settle/Simon	Mazoyer	$3n-2$	127	6	$\mathcal{O}(n^2)$	N/A	non	infini
Umeo	Minsky/McCarthy	$3n+\mathcal{O}(log(n))$	78	8	$\mathcal{O}(n^2)$	Réc. Bidir.	double	infini
Yunès	Minsky/McCarthy	$3n+\mathcal{O}(log(n))$	132	6	$\mathcal{O}(n.log(n))$	Réc. Bidir.	simple	fini
Yunès	Minsky/McCarthy	$4n+\mathcal{O}(log(n))$	195	8	$\mathcal{O}(n.log(n))$	Réc. Bidir.	simple	fini
Yunès	Minsky/McCarthy	$5n+\mathcal{O}(log(n))$	177	8	$\mathcal{O}(n.log(n))$	Réc. Bidir.	simple	fini
Yunès	Minsky/McCarthy	$4n+\mathcal{O}(log(n))$	161	7	$\mathcal{O}(n.log(n))$	Réc. Bidir.	simple	fini
Yunès	Minsky/McCarthy	$4n+\mathcal{O}(log(n))$	95	6	$\mathcal{O}(n^2)$	Réc. Bidir.	simple	fini
Yunès	Minsky/McCarthy	$6n+\mathcal{O}(log(n))$	136	6	$\mathcal{O}(n.log(n))$	Réc. Bidir.	simple	fini

TAB. 2.1 – Principales solutions au Firing Squad

2.5 Bibliographie

[1] J.-P. Allouche, V. Berthé. *Triangle de Pascal, complexité et automates.* Bulletin of Mathematical Belgian Society, Vol. 4, pp. 1–23. 1997.

[2] E. Astesiano, G. Reggio. *On the specification of the firing squad synchronization problem.* In proceedings, Workshop on the Analysis of Concurrent Systems, Cambridge, UK, LNCS 207, pp. 137–156, 1985.

[3] R.W. Baker, G.T. Herman. *CELIA - A Cellular Linear Iterative Array Simulator.* In proceedings, 4th annual conference on Applications of Simulation, New-York, NY, pp 64–73. 1970.

[4] R. Balzer. *An 8-State Minimal Time Solution to the Firing Squad Synchronization Problem.* Information and Control 10, pp. 22-42. 1967.

[5] A.P. Benkard. *Symmetries of the firing squad synchronization problem revealed in a nested array.* In proceedings, International conference on APL 1988, Sydney, Australia, pp. 19–27. 1988.

[6] E.R. Berlekamp, J.H. Conway, R.K. Guy. *Winning Ways for your Mathematical Plays. Second edition.* AK Peters. 2001. isbn 978-1568811307.

[7] A. Berthiaume, T. Bittner, L. Perkovic, A. Settle, J. Simon. *Bounding the Firing Squad Synchronization Problem on a Ring.* Theoretical Computer Science 320(2–3): 213–228, 2004.

[8] A. Berthiaume, L. Perkovic, A. Settle, J. Simon. *New Bounds for the Firing Squad Synchronization Problem on a Ring.* In proceedings, 9th International Colloquium on Structural Information and Communication Complexity, SIROCCO 9, Andros, Greece. pp. 17–31, 2002.

[9] A.W. Burks. *Essays on Cellular Automata.* University of Illinois Press, 1971.

[10] B.A. Coan, D. Dolev, C. Dwork, L.J. Stockmeyer. *The distributed firing squad problem.* STOC 1985, pp. 335–345. 1985.

[11] B.A. Coan, C. Dwork. *Simultaneity is harder than agreement.* Information and Computation, Vol. 91(2), pp. 205–231. April 1991.

[12] E.F. Codd. *Cellular Automata.* Academic Press, New York, 1968.

[13] K. Culik. *Variations of the firing squad problem and applications.* Information Processing Letters, Vol. 30(3), pp. 152–157. February 1989.

[14] K. Culik, S. Dube. *An efficient solution of the firing mob problem.* Theoretical Computer Science, 91(1), pp. 57–69. December 1991.

[15] J. Duprat. *Proof of correctness of the Mazoyer's solution of the firing squad problem in Coq.* `http://hdl.handle.net/2332/792`, 2002.

[16] J. Durand-Lose. *Calculer géométriquement sur le plan - machines à signaux.* Mémoire d'habilitation à diriger des recherches. Décembre 2003.

[17] M. Frandes, S. Verel, M. Clergue, P. Collard. *Towards a resolution of the firing squad problem with 5 states by metaheuristics.* 3ième rencontres FRAC, ANR Sycomore « Systèmes Complexes et modèles de calcul », Juin 2007, Nice, France. 2007.

[18] M. Garzon. *Models of Massive Parallelism : Analysis of Cellular Automata and Neural Networks.* Springer-Verlag, June 1995.

[19] H.-D. Gerken. *Über Synchronizations - Probleme bei Zellularautomaten.* Diplomarbeit, Institut für Theoretische Informatik, Technische Universität Braunschweig, 1987.

[20] J.-L. Giavitto. *Trois questions à Jean-Louis Giavitto.* Stic-Hebdo, n°42, Janvier 2005.
`http://www.ibisc.univ-evry.fr/~giavitto/asti-hebdo-42/sh42.html`

[21] D. Goldstein, K. Kobayashi. *On the complexity of network synchronization.* In International Symposium on Algorithms and Computation, ISAAC 2004. December 20–22, 2004. Journal version in SIAM J. Comput. Vol. 35(3) pp. 567–589. 2004.

[22] D. Goldstein, K. Kobayashi. *On the complexity of the "most general" firing squad synchronization problem.* In Proceedings, 23rd International Symposium on Theoretical Aspects of Computer Science, STACS 2006, February 23–25, 2006.

[23] D. Goldstein, N. Meyer. *The Wake Up and Report Problem is Time-Equivalent to the Firing Squad Synchronization Problem.* Distributed Computing 17(1), pp. 21–31, 2004.

[24] E. Goto. *A Minimum Time Solution of the Firing Squad Problem.* Course Notes for Applied Mathematics 298. Harvard University, 1962.

[25] A. Grasselli. *Synchronization of cellular arrays : the firing squad synchronization problem in two dimensions.* Information and Control, Vol. 28(2), pp. 113–124. June 1975.

[26] J.J. Grefenstette. *Network Structure and the Firing Squad Synchronization Problem.* Journal of Computer and System Sciences, Vol. 26(1), pp. 139–152. February 1983.

[27] P. Greussay. *L'ordinateur cellulaire.* La recherche, n°204, pp. 1320–1330, 1988.

[28] S. Grigorieff. *Synchronization of a Bounded Degree Graph of Cellular Automata with non Uniform Delays in Time* $\delta\lfloor log_m(\delta)\rfloor$. Theoretical Computer Science 356:170–185, 2006.

[29] J. Gruska, S. La Torre, M. Parente. *Optimal Time and Communication SOlutions of Firing Squad Synchronization Problem on Square Arrays, Toruses and Rings.* In proceedings, 8th International Conference in Developments in Language Theory, DLT 2004, Auckland, New Zealand, December 13–17, 2004, pp. 200-211. 2004.

[30] O. Heen. *Efficient Constant Speed-Up for one Dimensional Cellular Automata Calculators.* Parallel Computing, Vol. 23(11), pp. 1663–1671. 1997.

[31] O. Heen. *Linear Speed-Up for Cellular Automata Synchronizers and Applications.* Theoretical Computer Science, Vol. 188(1–2):45–47. 1997.

[32] G.T. Herman. *Synchronization of Growing Cellular Arrays.* Information and Control, 25(2). pp. 103–122. June 1974.

[33] G.T. Herman, W.H. Liu. *The daughter of CELIA, the French flag and the Firing Squad.* In proceedings. 6th conference on Winter simulation, San Fransico, CA. 1973.

[34] D. Hillis. *New Computer Architectures and Their Relationship to Physics or Why CS is No Good.* International Journal of Theoretical Physics, Vol. 21(3–4), pp. 255–262. 1982.

[35] D. Hillis. *The Connection Machine.* MIT Press. 1986. Partiellement disponible sur `books.google.fr`

[36] D. Hillis. *Richard Feynman and The Connection Machine.* Physics Today. Online version at
`http://www.longnow.org/views/essays/articles/ArtFeynman.php`

[37] K. Imai, K. Morita. *Firing Squad Synchronization Problem in Reversible Cellular Automata.* Theoretical Computer Science 165(2), pp. 475–482. October 1996.

[38] K. Imai, K. Morita, K. Sako. *Firing Squad Synchronization Problem in Number-Conserving Cellular Automata.* Fundamenta Informaticæ 52(1–3):133–141, 2002.

[39] T. Jiang. *The Synchronization of Non-Uniform Networks of Finite Automata.* Information and Control 97, pp. 234–261, 1992.

[40] D.E. Knuth. *The Art of Computer Programming.* Vol. 1, 2 & 3. Addison-Wesley. 1998. isbn 978-0201485417.

[41] K. Kobayashi. *The Firing Squad Synchronization Problem for Two Dimensional Arrays.* Information and Control, 34. pp. 177–194. 1977.

[42] K. Kobayashi. *The Firing Squad Synchronization Problem for a Class of Polyautomata Networks.* Journal of Computer and System Sciences 17:300–318, 1978.

[43] K. Kobayashi. *On time optimal solutions of the firing squad synchronization problem for two-dimensional paths.* Theoretical Computer Science 259(1–2), pp. 129–143, May 2001.

[44] K. Kobayashi. *A Complexity-Theoretical Approach to the Firing Squad Synchronization Problem.* Journées de l'Informatique Messine - NP-Completeness and Parallelism, May 17–18, Institute of Technology, Metz, France. 1999.

[45] K. Kobayashi, D. Goldstein. *On formulations of Firing Squad Synchronisation Problems.* In Proceedigns, 4th International Conference on Unconventional Computation, UC2005. October 3–7, 2005.

[46] M. Kutrib, R. Vollman. *Minimal time synchronization in restricted defective cellular automata.* Journal of Information Processing and Cybernetics, EIK 27:179–196, 1991.

[47] M. Kutrib, R. Vollmar. *The Firing Squad Synchronization Problem in Defective Cellular Automata.* IEICE Transactions on Information and Systems E78-D:895–900, 1995.

[48] S. La Torre, M. Napoli, M. Parente. *Firing Squad Synchronization Problem on Bidimensional Cellular Automata with Communication Constraints.* In proceedings, M. Margenstern, Y. Rogozhin eds, 3rd

International Conference on Machines, Computations and Universality, MCU 2001, Chisinau, Moldavia, May 23–27, 2001, pp. 264–275. 2001.

[49] L. Levin. *Theory of computation : Fundamentals of computing* . Lecture notes.
`http:www.cs.bu.edu/fac/lnd/toc/`

[50] O. Martin, A. Odlyzko, S. Wolfram. *Algebraic Properties of Cellular Automata.* Communications in Mathematical Physics, Vol. 93, p. 219. 1984.

[51] J. Mazoyer. *A Six-State Minimal Time Solution to the Fring Squad Synchronization Problem.* Theoretical Computer Science, 50:183–238, 1987.

[52] J. Mazoyer. *A Minimal Time Solution to the Firing Squad Synchronization Problem with Only One Bit of Information Exchanged.* Rapport Technique LIP 89.03, École Normale Supérieure de Lyon, 1989.

[53] J. Mazoyer. *A minimal-time solution to the FSSP without recursive call to itself and with bounded slope of signals.* Technical Report, Draft Version. September 1998.

[54] J. Mazoyer, V. Terrier. *Signals in One-Dimensional Cellular Automata.* Theoretical Computer Science, 217(1): 53–80, 1999.

[55] J. Mazoyer. *On Optimal Solutions to the Firing Squad Synchronization Problem.* Theoretical Computer Science, 168(2):367–404, 1996.

[56] M. Minsky. *Computation : Finite and Infinite Machines.* Prentice-Hall, 1967.

[57] E.F. Moore. *The Firing Squad Synchronization Problem* in *Sequential Machines. Selected Papers.* (E.F. Moore, eds.) Addison-Wesley, Reading MA, pp. 213–214, 1964.

[58] E.F. Moore, C.E. Shannon. *Reliable Circuits using less reliable Relays.* Journal of the Franklin Institute, pp. 191–208, 281–297. 1956.

[59] F.R. Moore and G.G. Langdon. *A generalized firing squad problem.* Information and Control, Vol. 12(3), pp. 212–220. March 1968.

[60] H. Nishio. *Fix a Local Function and Change Neighborhoods.* 13th International Workshop on Cellular Automata, AUTOMATA 2007. Fields Institute, Toronto Canada. August 27–29, 2007.

[61] H. Nishio. *Changing the Neighborhood of Cellular Automata.* In proceedings, J. Durand-Lose, M. Margenstern eds, 5th International Conference on Machines, Computations and Universality, MCU 2007, pp. 255–266. Orléans, France, September 10–14, 2007.

[62] Y. Nishitani, N. Honda. *The Firing Squad Synchronization Problem for Graphs.* Theoretical Computer Science, 14(1), pp. 39–61. 1981.

[63] K. Noguchi. *Simple 8-state minimal time solution to the firing squad synchronization problem.* Theoretical Computer Science 314:303–334, 2004.

[64] E.T. Ordman. *Byzantine Firing Squad using a Faulty External Source.* WDAG 1987, pp. 76–83. 1987.

[65] L. Pierre. Private communication.

[66] N. Pippenger. *Developments in "The Synthesis of Reliable Organisms from Unreliable Components".* In proceedings, Legacy of J. vin Neumann, Symposia in Pure Mathematics, Vol. 50, pp. 311–324. 1990.

[67] J.M. Rifflet. *Algorithmique et Programmation Répartie : Étude de quelques problèmes liés à la répartition.* Notes de cours de Master 2, Université Paris 7, Denis Diderot. 2005.

[68] F. Robert. *Itérations Discrètes : une étude métrique.* Publications de l'Université de Grenoble.

[69] Z. Róka. *The Firing Squad Synchronization Problem on Cayley Graphs.* Theoretical Computer Science 244(1-2):243–256, 2000.

[70] F. Romani. *Cellular Automata Synchronization.* Information Sciences 10(2):299–318, 1976.

[71] F. Romani. *On the fast synchronization of tree connected networks.* Information Sciences 12(3):229–244, 1977.

[72] F. Romani. *The parallelism principle : speeding up the cellular automata synchronization.* Information and Control, Vol. 36(3), pp. 245–255. March 1978.

[73] P. Rosenstiehl, J. Fiksel & A. Holliger. *Intelligent Graphs : Networks of Finite Automata Capable of Solving Graph Problems.* in Graph Theory and Computing (R.C. Read ed.), Academic Press. 1972.

[74] P. Sanders. *Massively parallel search for transition-tables of polyautomata.* In : C. Jesshope, V. Jossifov, W. Wilhelmi (Eds), Proc. of

the VI Int. Workshop on Parallel Processing by Cellular Automata and Arrays, Akademie, Berlin, pp. 99-105. 1994.

[75] P. Sarkar. *A brief histor of cellular automata.* ACM Computing Surveys (CSUR) Vol. 32, Issue 1, pp. 80–107. March 2000.

[76] J. Savage. *Models of Computation : Exploring the Power of Computing.* Addison-Wesley, Berkeley, CA. 1998.

[77] H. Schmid. *Synchronisationsprobleme für zelluläre Automaten mit mehreren Generälen.* Diplomarbeit. September 2003.

[78] H. Schmid, T. Worsch. *The Firing Squad Synchronization Problem with Many Generals for One-Dimensional CA.* In proceedings, J.-J. Levy, E. W ; Mayr, J. C. Mitchell eds, 18th World Computer Congress, 3rd International Conference on Theoretical Computer Science, TCS 2004, 22–27 August, 2004, Toulouse, France, pp. 111–124. 2004.

[79] A. Settle, J. Simon. *Non-minimal Time Solutions for the Firing Squad Synchronization Problem.* Technical Report 97-08, University of Chicago. 1997.

[80] A. Settle, J. Simon. *Improved Bounds for the Firing Squad Synchronization Problem.* In proceedings, 5th International Colloquium on Structural Information and Communication Complexity, Amalfi, Italy. pp. 66–81. 1998.

[81] A. Settle, J. Simon. *Smaller Solutions for the Firing Squad.* Theoretical Computer Science, 276(1):83–109, 2002.

[82] I. Shinahr. *Two and Three Dimensional Firing Squad Synchronization Problem.* Information and Control 24(2), pp. 163–180. February 1974.

[83] H. Szwerinski. *Time-optimal Solution of the Firing Squad Synchronization Problem for n-Dimensional Rectangles with the General at an Arbitrary Position.* Theoretical Computer Science 19:305–320, 1982.

[84] H. Umeo. *A fault-tolerant scheme for optimum-time firing squad synchronization.* In proceedings of PARCO 1993, pp. 223–230. 1993.

[85] H. Umeo. *An Efficient Design of Two-Dimensional Firing Squad Synchronization Problem.* Eighth International Workshop on Cellular Automata, Prague, Czechia. 2002.

[86] H. Umeo. *A Simple Design of Time-Efficient Firing Squad Synchronization Algorithms with Fault-Tolerance.* IEICE Transactions on Information and Systems E87-D(3), 2004.

[87] H. Umeo. *A Note on Firing Squad Synchronization Algorithms.* Schloss Rauischholzhausen, Workshop on Cellular Automata, IFIP TC1 WG1.5, Rauischholzhausen, Germany, March 1996.

[88] H. Umeo, M. Hisaoka, T. Sogabe. *An Investigation into Transition Rul Sets for Optimum-time Firing Squad Synchronization Algorithms on One-Dimensional Cellular Automata.* Interdisciplinary Information Sciences, Vol. 8(2), pp. 207–217. 2002.

[89] H. Umeo, M. Hisaoka, T. Sogabe. *A comparative study of optimum-time synchronization algorithms for one dimensional cellular automata : a survey.* In proceedings, M.A.P. Sloot, B. Chopard, G.A. Hoekstra eds, 6th International Conference on Cellular Automata for Research and Industry, Amsterdam, Netherland, 25–27 October 2004, pp. 50–60. LNCS 3305, 2004.

[90] H. Umeo, M. Maeda, N. Fujiwara. *An efficient mapping scheme for embedding any one-dimensional firing squad synchronization algorithm onto two-dimensional arrays.* In proceedings, Cellular Automata 5th International Conference on Cellular Automata for Research and Industry, ACRI 2002, Geneva, Switzerland, October 9–11 2002, pp. 69–81. 2002.

[91] H. Umeo, M. Maeda, M. Hisaoka, M. Teraoka. *A state-efficient mapping scheme for designing two-dimensional firing squad synchronization algorithms.* Fundamenta Informaticæ 74(4):603–623, 2006.

[92] H. Umeo, M. Maeda, K. Hongyo. *A Design of Symmetrical Six-State 3n-Step Firing Squad Synchronization Algorithms and Their Implementations.* Proceedings of ACRI 2006, LNCS 4173, pp 157–168. 2006.

[93] H. Umeo, T. Sogabe, Y. Nomura. *Correction, Optimization and Verification of Transition Rule Set for Waksman's Firing Squad Synchronization Algorithm.* In proceedings, S. Bandini and T. Worsch eds, Theoretical and Practical Issues on Cellular Automata, 4th International Conference on Cellular Automata for Research and Industry, ACRI 2000, Karlsruhe, Germany, 4–6 October 2000. Springer, 2000. isbn 1-85233-388-X.

[94] H. Umeo, T. Yanagihara. *A smallest five-state solution to the firing squad synchronization problem.* In proceedings, J. Durand-Lose, M. Margenstern eds, 5th International Conference on Machines, Computations and Universality, MCU 2007, pp. 291–302. Orléans, France, September 10–14, 2007.

[95] H. Umeo, T. Yanagihara, M. Kanazawa. *State-Efficient Firing Squad Synchronization Protocols for Communication-Rectricted Cellular Automata.* Proceedings of ACRI 2006, LNCS 4173, pp 169–191. 2006.

[96] H. Umeo, H. Yanase, M. Hisaoka. *One-Sided Recursive Synchronization Algorithms for One-Dimensional Cellular Arrays.* Memoirs of Osaka Electro-Communication University. Natural Science, Vol. 40, pp. 37–45. 2005.

[97] H. Vivien. *Cellular Automata : a geometrical approach.* Book to appear.

[98] H. Vivien. *A quasi-optimal time for synchronizing two interacting finite automata.* Journal of Algebra and Computation, Vol. 6(2), pp. 261–267. 1996.

[99] R. Vollmar. *Yet another generalization of the firing squad problem.* Technical report, Technische Universität Braunschweig, Braunschweig, 1976.

[100] R. Vollmar. *Some remarks about the "efficiency" of polyautomata.* International Journal of Theoretical Physics, Vol. 21, pp. 1007–1015. 1982.

[101] J. von Neumann. *Collected Works. Volume V. Design of Computers, Theory of Automata and Numerical Analysis.* Pergamon Press. 1963.

[102] J. von Neumann. *Probabilistics logics and the synthesis of reliable organisms from unreliable components.* In C.E. Shannon and J. McCarthy eds, Automata Studies, pp. 43–98. 1955.

[103] A. Waksman. *An Optimum Solution to the Firing Squad Synchronization Problem.* Information and Control 9, pp. 66–78. 1966.

[104] S. Wolfram. *Cellular Automata and Complexity : Collected Papers.* Westview Press, 2002.

[105] S. Wolfram. *A new kind of science.* Wolfram Media, 2002.

[106] T. Worsch. *How to achieve universality in a CA using the same local rule but different neighborhoods.* 13th International Workshop on

Cellular Automata, AUTOMATA 2007. Fields Institute, Toronto Canada. August 27–29, 2007.

[107] J.-B. Yunès, *Synchronisation et Automates Cellulaires : la ligne de fusiliers.* (PhD thesis) Thèse LITP 93/01, 1993.

[108] J.-B. Yunès. *Seven States Solutions to the Firing Squad Synchronization Problem.* Theoretical Computer Science, 127(2):313-332, 1994.

[109] J.-B. Yunès. *Fault Tolerant Solutions to the Firing Squad Synchronization Problem.* Dagstuhl seminar 9510, Germany. 1995.

[110] J.-B. Yunès. *Fault Tolerant Solutions to the Firing Squad Synchronization Problem in Linear Cellular Automata.* Journal of Cellular Automata 1(3):253–268, 2006.

[111] J.-B. Yunès. *Revisiting existing solutions to the firing squad synchronization problem.* Workshop on Symbolic Dynamic and Coding. Marne-la-vallée, France, July 2–4, 2007.

[112] J.-B. Yunès. *New extensions to some firing squad synchronization solutions.* 13th International Workshop on Cellular Automata, AUTOMATA 2007. Fields Institute, Toronto Canada. August 27–29, 2007.

[113] J.-B. Yunès. *Simple new algorithms which solve the FSSP.* In proceedings, J. Durand-Lose, M. Margenstern eds, 5th International Conference on Machines, Computations and Universality, MCU 2007, pp. 316–324. Orléans, France, September 10–14, 2007.

[114] J.-B. Yunès. *An Intrinsically non Minimal-Time Minsky-like 6-States Solutions to the Firing Squad Synchronization Problem.* To appear in RAIRO ITA/TIA. 2008.

[115] J.-B. Yunès. *Cellular Automata Synchronizers Insensitive to some Initial Conditions.* Submited to Theoretical Computer Science. 2007.

[116] J.-B. Yunès. *Known CA synchronizers made insensitive to the initial state of the initiator.* Submitted to Journal of Cellular Automata. 2007.

[117] J.-B. Yunès. *A 4-states algebraic solution to linear cellular automata synchronization.* Submitted to Information Processing Letters. 2007.

[118] J.-B. Yunès. *About linear-time synchronization of cellular automata.* In preparation to answer a Special Call for Papers in FI issued after MCU'07. 2008.

[119] J.-B. Yunès. *A simple Goto-like solution to the FSSP.* In preparation. 2007.

[120] J.-B. Yunès. *CA - LFSSP Explorer.* `MacOSX®` version. `http://www.liafa.jussieu.fr/~yunes/ca/`

Chapitre 3

Travaux sur les fonctions Booléennes

3.1 Une tentative de cryptanalyse : HFE

La première partie de mes travaux sur les fonctions booléennes s'inscrit dans une tentative de cryptanalyse d'un système de chiffrement à clé publique connu sous le nom de HFE Hidden Field Equation (voir [42, 52, 53]).

Un chiffrement à clé publique est similaire au mécanisme de transmission de courrier par boîte aux lettres : tout un chacun peut déposer un courrier dans une boîte pourvu que l'on connaisse la boîte, mais seul le destinataire peut en retirer le courrier avec sa clé. Le secret d'une telle transmission réside dans la difficulté à pouvoir ouvrir la boîte sans la clé. Bien que simple, ce mécanisme n'avait pas de version mathématisée connue[1] avant les travaux de Diffie et Hellman (voir [16].

De façon plus formelle, un cryptosystème à clé publique est une paire de fonctions (réalisées par un algorithme et notées ici f, g) qui permet à tout un chacun de chiffrer un message (noté M) à l'aide d'une clé connue de tous (clé publique notée c). On chiffre un message (noté M') de la façon suivante : $M' = f_c(M)$, où f_c est une fonction entièrement déterminée par la fonction

[1] C'est un des problèmes de la cryptographie : elle a longtemps été une science interdite réservée aux militaires. Nombre de secrets cryptographiques ont été bien gardés et un certain nombre le sont probablement encore ; toutefois en raison des progrès réalisés par des chercheurs non-militaires ces 25 dernières années, certains de ces secrets sont déclassifiés. Pour Diffie-Hellman on a récemment appris que les britanniques et les américains connaissaient des méthodes analogues - voir [19, 65].

f et la clé c.
Lorsque les fonctions f, g sont bien conçues il est très difficile de retrouver le message M à partir de M', de f, g et de c. Lorsqu'on dit difficile, c'est juste difficile... Cette notion est donc toute relative car la difficulté à « inverser » peut être liée à la durée de vie des secrets échangés.
Le destinataire du message, possède lui une clé connue de lui seul (clé privée, notée c') qui lui permet d'inverser le chiffrement (on parle de déchiffrement) par une opération similaire : $M = g_{c'}(M')$, où g est une fonction entièrement déterminée par la clé c'.

On dit généralement que la cryptographie s'occupe de concevoir des systèmes de chiffrement-déchiffrement et que la cryptanalyse s'occupe de casser ces systèmes, c'est-à-dire de trouver leurs failles.

3.1.1 HFE

Les cryptosystèmes HFE sont construits à l'aide de polynômes définis sur des corps finis. La méthode peut être décrite ainsi :

Prenons un corps $\mathbb{K}$, extension de degré n de $\mathbb{F}_2$. Alors $\mathbb{K}$ peut être vu comme un espace vectoriel de dimension n sur $\mathbb{F}_2$. Toute base $e_1, \ldots, e_n$ de $\mathbb{K}$ définit une bijection de $\mathbb{K}$ sur $(\mathbb{F}_2)^n$ de la façon suivante :

$$\sum_{i=1}^{n} x_i.e_i \longleftrightarrow (x_1, \ldots, x_n), \text{ where } x_i \in \mathbb{F}_2$$

Ainsi n'importe quel polynôme monovarié quasi-quadratique

$$P(X) = \sum_{i,j} \alpha_{i,j} X^{2^i+2^j} + \gamma \text{ avec } \alpha_{i,j} \in \mathbb{K} \text{ et } \gamma \in \mathbb{K}$$

sur $\mathbb{K}$ peut être vu dans $(\mathbb{F}_2)^n$ comme un ensemble de polynômes multivariés quasi-quadratiques sur $\mathbb{F}_2$:

$$\left\{ \begin{array}{rcll} P_1(x_1, \ldots, x_n) & = & \sum_{j,k}^{1..n} \beta_{1,j,k} x_i x_k + \delta_1 & \\ & \vdots & & \text{avec } \beta_{i,j,k} \in \mathbb{F}_2 \text{ et } \delta_i \in \mathbb{F}_2 \\ P_n(x_1, \ldots, x_n) & = & \sum_{j,k}^{1..n} \beta_{n,j,k} x_i x_k + \delta_n & \end{array} \right.$$

La donnée d'un tel système constitue la clé publique d'un système HFE. Le chiffrement se fait par simple application des polynômes P_i sur les bits du texte clair. Quant au déchiffrement, il est supposé être NP-hard (voir

[25, 33, 15, 31]). Toutefois on sait que si le degré de $P(X)$ n'est pas trop élevé (voir [64]) l'algorithme de Berlekamp (voir [2, 34]) fonctionne en temps raisonnable.

Pour garantir l'inviolabilité, il est introduit deux « portes cachées » qui sont de simples permutations de $\mathbb{K}$ et que nous appellerons S et T (on notera que leur expression sur $\mathbb{K}$ est un polynôme de la forme $\sum_i \alpha_i X^{2^i}$ et sur $\mathbb{F}_2$ une transformation linéaire inversible). Alors la clé publique HFE est définie par l'expression sur $\mathbb{F}_2$ du polynôme $T(P(S(X)))$ (qui reste quasi-quadratique). Mais désormais le polynôme $T(P(S(X)))$ est de degré bien trop élevé pour pouvoir espérer utiliser l'algorithme de Berlekamp pour en trouver les racines. La clé privée est alors constituée de la donnée de S et T, le polynôme pouvant être rendu public.

3.1.2 Une tentative de cryptanalyse

Notre attaque (voir [48]) est à ranger dans la classe des attaques à texte chiffré seul où l'on suppose n'avoir à sa disposition que la clé publique et un exemplaire du texte chiffré. Nous avons décidé de ne pas tenter d'analyse algébrique bien que le système le soit, mais de feindre l'ignorance en prenant le système tel qu'il se présente : un système d'équations à variables booléennes. Pour les cryptanalyses algébriques on pourra se reporter aux travaux de Nicolas Courtois (voir [15]), Aviad Kipnis et Adi Shamir (voir [33]), Jean-Charles Faugère (voir [22]) et Louis Granboulan, Antoine Joux et Jacques Stern (voir [26]).

Notre idée fut d'utiliser les BDDs pour représenter les équations et résoudre le système. Les BDDs sont une représentation des fonctions booléennes qui permet de les manipuler de façon efficace.

Ainsi, étant donné un texte chiffré $y_1, \ldots, y_n$ le système de HFE nous permet d'écrire que

$$\begin{cases} y_1 &= P_1(x_1, \ldots, x_n) \\ & \vdots \\ y_n &= P_n(x_1, \ldots, x_n) \end{cases}$$

Il nous suffit donc de satisfaire la formule (c'est-à-dire trouver des booléens $x_1, \ldots, x_n$ qui satisfont la formule) :

$$F(x_1, \ldots, x_n) \stackrel{def}{=} \bigwedge_{i=1}^{n} (1 + y_i + P_i(x_1, \ldots, x_n))$$

pour extraire le message clair. Cette formule étant construite en utilisant les BDD et par itération en construisant

$$F_{i+1} = F_i \bigwedge (1 + y_i + P_i(x_1, \ldots, x_n))$$

où $F_0 = 1$ et $F = F_n$.

Malheureusement, il nous fut impossible ne serait-ce que de représenter sous la forme d'un BDD une seule équation d'un système HFE utilisable en pratique (80 variables booléennes selon la pratique standard[2]).

L'échec est dû à la « complexité » des fonctions booléennes engendrées dans un système HFE. Pire, les expériences menées ensuite ont montré que la représentation BDD ne distinguait même pas les systèmes HFE des équations quasi-quadratiques tirées au hasard. Le lecteur est invité à lire [17] pour obtenir un tel résultat.

Nous avons donc choisi de tenter de fabriquer des systèmes HFE plus petits afin de réaliser des cryptanalyses complètes selon notre méthode afin de tenter de comprendre les phénomènes en jeu. Pour cela nous avons du fabriquer deux logiciels. L'un pour construire des systèmes HFE selon notre convenance (voir [49]), et l'autre pour manipuler efficacement les BDD engendrés (voir [68]).

On trouvera en figure 3.1 un exemple de résultats obtenus. Dans ce diagramme différentes courbes apparaissent. Chacune représente un « comportement » typique de notre algorithme de résolution de système pour un système HFE produit à partir d'un corps fini particulier. Ces courbes mesurent la taille du BDD de chacune des formules F_i générées.

De façon évidente, les valuations qui satisfont la formule F, satisfont aussi toute formule F_i. Pourquoi la courbe a t-elle une forme pareille ? Ce qu'il faut comprendre c'est qu'à chaque étape le nombre de valuations qui satisfont la formule F_i diminue, mais la « complexité descriptive » de cet ensemble de solution varie selon la courbe. Au début, la complexité croit fortement, puis décroît fortement jusqu'à s'effondrer très rapidement vers un BDD minuscule (la solution se décrit très facilement). Ce que l'on peut donc observer c'est que la taille des BDDs engendrés passe par un pic. Quel est la taille de ce pic ? À quoi correspondent les fonctions cachées derrière ce pic ? Ce sont les questions que nous nous sommes posées, d'où notre intérêt pour les BDDs en tant qu'objet d'étude.

[2]Ceci afin de résister à ce que l'on appelle une attaque par force brute. En effet tester 2^{80} chiffrements, donc 2^{79} en moyenne, est considéré comme une limite difficilement atteignable, encore qu'avec Internet...

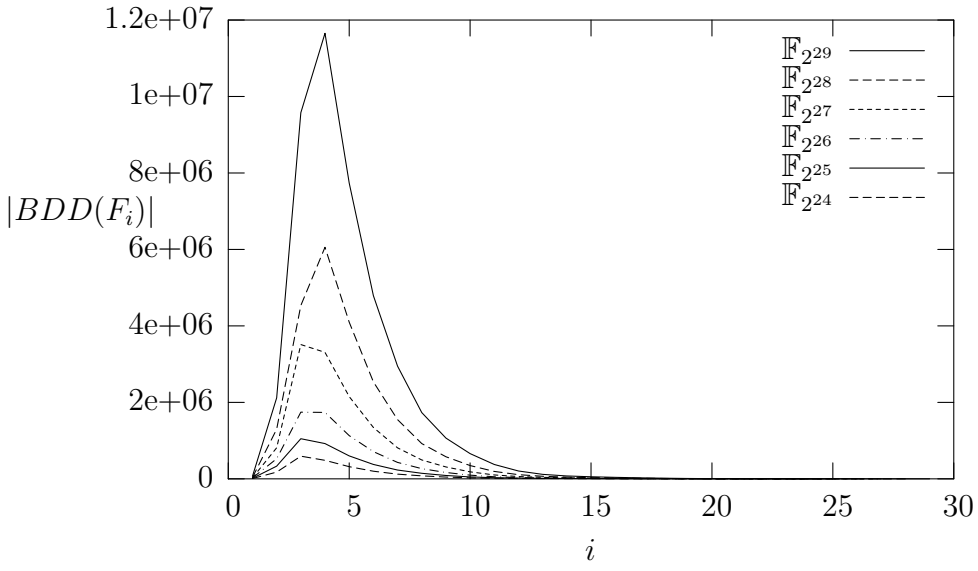


FIG. 3.1 – Taille des BDDs engendrés par l'algorithme de résolution

3.2 Les BDDs

Les diagrammes de décision binaires (BDD - Binary Decision Diagrams) sont des représentations des fonctions booléennes. Pour de nombreuses fonctions utiles, cette représentation est concise ce qui permet la manipulation efficace des fonctions booléennes par leur intermédiaire. Cette structure est particulièrement employée dans le domaine de la vérification. Il n'est pas dans notre but d'exposer ici le nombre important des variantes existantes de cette représentation, et nous renvoyons le lecteur aux œuvres de Randal Bryant (voir [4] et son site web) ou de Ingo Wegener (voir [61, 62]).

Les QROBDDs (Quasi Reduced Ordered BDD) sont des représentations canoniques des fonctions booléennes. Un QROBDD est construit de la façon suivante : partant de la table de vérité d'une fonction booléenne donnée f, on construit l'arbre binaire qui lui est canoniquement associée et dont les feuilles sont étiquettées par `0` ou `1`. Alors en identifiant tous les sous-graphes isomorphes on obtient un graphe acyclique dirigé que l'on appelle $QROBDD(f)$. Un exemple est donné en figure 3.2.

Il faut noter, que les QROBDD sont une représentation canonique des

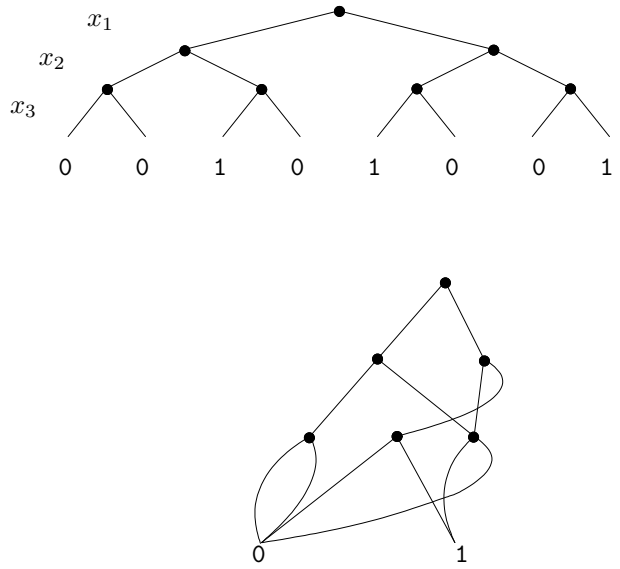


FIG. 3.2 – Obtention d'un QROBDD

fonctions booléennes pas les ROBDD habituellement considérés[3]. En effet et par exemple, pour tout k entier, les fonctions $f(x_1, \ldots, x_k) = 1$ ont toutes la même représentation ROBDD, mais des représentations QROBDD toutes différentes (« fil » de hauteur k).

La complexité $c_{\text{qrobdd}}(f)$ d'une telle représentation vérifie la propriété triviale suivante :

$$n + 1 \leq c_{\text{qrobdd}}(f) \leq 2^n + 1$$

Si la borne minimale est bien la bonne, il n'en est rien de la borne maximale dont la valeur, notons la $C(n)$, a beaucoup été étudiée (voir [27, 28, 29]). Notre but fut alors de décrire l'ensemble des fonctions booléennes à n variables ayant pour complexité la valeur maximale. Ces fonctions seront désormais qualifiées de « dures ».

3.2.1 Le profil d'un BDD

À quoi un BDD ressemble t-il ?

[3]On sur-ajoute aux QROBDD une opération de réduction qui supprime les nœuds conduisant à deux sous-graphes isomorphes.

Pour répondre à cette question il faut d'abord remarquer qu'un QROBDD est étagé, *i.e.* qu'il est constitué de nœuds chacun situé à une certaine distance de la racine. Nous appellerons niveau i d'un QROBDD, l'ensemble des nœuds situés à distance i de la racine. Nous noterons $\mathcal{P}(f)$ le profil du QROBDD associé à f, $\mathcal{P}(f) = (r_0(f), \ldots, r_n(f))$ où $r_i(f)$ est le nombre de nœuds du niveau i du QROBDD associé à f. Les relations suivantes sont connues : $r_0(f) = 1$, $r_n(f) = 1$ ou 2, $r_{i+1}(f) \leq 2.r_i(f)$ et $r_i(f) \leq r_{i+1}(f)^2$.

Par définition nous avons

$$c_{\text{qrobdd}}(f) = \sum_{i=0}^{n} r_i(f)$$

Ce qu'il nous reste à déterminer, c'est comment les nœuds d'un étage donné sont-ils connectés aux nœuds de l'étage suivant ?

Pour le déterminer nous avons établi un résultat de combinatoire indiquant de façon constructive que le nombre de façons de connecter un étage de k nœuds à un étage de m nœuds et que nous notons (par analogie avec les cœfficients binomiaux) $C(m, k)$ vérifie la relation

1. $C(0, 0) = 1$
2. if $k > m^2$ or $k < \frac{m}{2}$ then $C(m, k) = 0$
3. else
$$C(m, k) = \binom{m^2}{k} - \sum_{j=1}^{m} \binom{m}{j} C(m - j, k) > 0$$

où encore sous forme d'une sorte de triangle de Pascal

1. $C(0, 0) = 1$, $\forall\, p > 0$, $C(0, p) = C(p, 0) = 0$,
2. $C(1, 1) = 1$, $\forall\, p > 1$, $C(1, p) = 0$,
3. $\forall\, m > 1$, $\forall\, k > 0$, then

$$\begin{aligned} kC(k, m) = {} & (m^2 - k + 1)C(k - 1, m) \\ & + m(2m - 1)C(k - 1, m - 1) \\ & + m(m - 1)\ C(k - 1, m - 2) \end{aligned}$$

Ainsi, le nombre de BDD ayant le profil $(r_0, \ldots, r_n)$ est égal à $\prod_{i=1}^{n} C(r_i, r_{i-1})$.

Ce que nous n'avons pas pu encore déterminer est pour une complexité donnée, quels sont les profils possibles et par là le nombre de QROBDDs (donc de fonctions) ayant cette complexité.

Pour des valeurs particulières des profils, le nombre de BDD peut-être facilement déterminé et les fonctions associées caractérisées. En particulier pour les BDDs de taille maximale.

Une remarque importante, l'étude faite ne consiste en réalité quà dénombrer des graphes particuliers que nous avons appelés graphes booléens, car les résultats obtenus n'ont finalement rien à voir avec les fonctions booléennes On notera au passage que ces résultats peuvent être aisément étendus à d'autres graphes de ce type (quadtrees et autres octrees utilisées en infographie voire en compression).

Maintenant quels rapports y a t-il entre ces graphes et les fonctions booléennes ? Ce rapport n'est pas complètement clair, mais si l'on fixe une décomposition des fonctions booléennes (comme la décomposition de Shannon) alors on associe à un graphe donné une fonction booléenne particulière. Donc en fixant la décomposition de Shannon (traditionnellement utilisée pour obtenir un BDD), la question est de déterminer l'ensemble des fonctions booléennes dont la décomposition de Shannon donne un graphe booléen de taille maximale. Mais la question pourrait se poser pour d'autres décompositions.

3.3 Les fonctions booléennes dures

Pour la décomposition de Shannon, le résultat que nous avons obtenu est que les fonctions booléennes dures sont toutes liées à la fonction Storage Access (notée SA) qui est la plus simple parmi les fonctions dures. Si on note $\mathcal{H}_n$ l'ensemble des fonctions dures à n variables, alors pour certaines valeurs de n (*i.e.* lorsque $n = a + 2^a$, a entier) alors l'ensemble $\mathcal{H}_n$ est exactement l'ensemble des Twisted-SA, c'est-à-dire l'ensemble des fonctions SA « perturbées » par l'action d'un groupe symétrique.

La fonction SA_k, pour tout $k = 2^a$, est une fonction à $n = a+2^a$ variables et est définie de la façon suivante :

$$SA_k(x_0, \ldots, x_{2^a-1}, y_0, \ldots, y_{a-1}) = x_m$$

où m est l'entier dont le développement en base 2 est $y_0, \ldots, y_{a-1}$. On peut l'interpréter comme une fonction qui représente les accès mémoire, la suite de bits x_i est l'état de la mémoire, et les y_i l'adresse de la mémoire dont on veut obtenir le contenu.

Cette fonction est connue pour avoir une représentation BDD de taille maximale (voir [62]). Attention, car si l'on change l'ordre des variables, pour certains ordres la représentation n'est plus du tout maximale (on peut même obtenir des représentations polynomiales). Toutefois, changer l'ordre des variables, c'est pour nous changer la fonction, il ne s'agit donc plus de la même fonction. Si elles sont liées (l'ordre des variables les lient), nous ne pouvons en déduire qu'il s'agit d'une même fonction.

Nous définissons ensuite la fonction Σ-twisted SA, pour $\Sigma \in \mathfrak{S}_{2^k}$ permutation de 2^k éléments induisant une bijection sur l'ensemble des k-uplets de bits représentant les entiers de 0 à $2^k - 1$, ainsi :

$$SA_k^{\Sigma}(x_0, \ldots, x_{2^a-1}, y_0, \ldots, y_{a-1}) = SA_k(\Sigma(X), y_0, \ldots, y_{a-1})$$

où X est l'entier dont le développement en binaire est $x_0 \cdots x_{k-1}$.

Nous montrons premièrement que les fonctions SA_k sont dures. Puis en second temps que $\mathcal{H}_{a+2^a} = \{SA_{2^a}^{\Sigma}\}$.

L'autre résultat obtenu est que pour les valeurs non spéciales de n ($a + 2^a < n < a+1+2^{a+1}$) l'ensemble des fonctions dures à n variables est obtenu par des injections à partir des SA_{2^a} et $SA_{2^{a+1}}$.

Ces résultats sont établis en constatant que le seul degré de liberté existant dans les QROBDD maximaux réside dans les connexions apparaissant au niveau d'inflexion (étage à partir duquel le nombre de nœuds diminue), et que c'est donc en cet endroit qu'apparaît l'action du groupe symétrique.

Des propriétés intéressantes ont été observées. Par exemple qu'il existe des fonctions dures qui ne dépendent pas de toutes leurs variables. C'est tout à fait surprenant car contraire à l'intuition. Pour en savoir davantage, nous reportons le lecteur à la lecture de [47].

3.4 Plus loin

Une étude des profils des graphes booléens reste à faire.

Ensuite, nous pouvons envisager étudier d'autres familles de fonctions booléennes, quelles sont les fonctions booléennes qui n'ont pas la complexité maximale mais presque ?

Plus loin, quels rapports y a t-il entre les BDDs et les SAT-solver ?

Pour revenir à notre première préoccupation, y a t-il un usage intéressant des BDDs en cryptanalyse. Nous pensons que c'est bien le cas, si notre tentative a échouée, nous n'y avons pas renoncé, car il reste bien d'autre angles

d'attaques en s'aidant des BDDs. Et il existe tout de même des résultats intéressants sur l'utilisation des BDDs en cryptanalyse (voir [36]).

Sinon, songeons au graphe de transition des configurations d'une machine de Turing, c'est-à-dire le graphe qui lit deux configurations d'une machine de Turing s'il existe une transition de la machine permettant de passer de l'un à l'autre. Ce graphe peut être représenté par une fonction booléenne à nombre d'argument infini. Si l'on songe à représenter cette fonction sous la forme d'un graphe booléen infini, que ce graphe booléen infini représente un ensemble de di-adiques, quel rapport y a t-il entre les di-adiques et les machines de Turing ? Si l'on regarde le graphe booléen d'une machine universelle, quel est-il ? Et avec un automate cellulaire ?

3.5 Bibliographie

[1] S.B. Akers. *Binary Decision Diagrams.* Transactions on Computers. C-27(6), pp. 509–516. 1978.

[2] E.R. Berlekamp. *Factoring Polynomials Over Finite Fields.* Bell Systems Technical Journal, Vol. 46, pp. 1853–1859. 1967.

[3] B. Bollig, I. Wegener. *Improving the variable ordering of OBDDs is NP-complete.* IEEE Transactions on Computers, Vol. 45(9), pp. 993–1002. 1996.

[4] R.E. Bryant. *Graph-based algorithms for boolean functions.* In IEEE Transactions on Computers. C-35(8), pp. 677–691. 1986.

[5] R.E. Bryant. *On the complexity of VLSI and graph representations of boolean functions with applications to integer multiplication..* In IEEE Transactions on Computers, Vol. 40(2), pp. 205–213. 1991.

[6] B. Buchberger. *A Theoretical Basis for the Reduction of Polynomials to Canonical Forms.* ACM SIGASM Bulletin 10/3, pp. 19–24. 1976.

[7] B. Buchberger. *Some properties of Gröbner Bases for Polynomial.* ACM SIGASM Bulletin 10/4, pp. 19–29. 1976.

[8] B. Buchberger. *A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases.* In proceedings, EUROSAM'79 Symposium on Symbolic and Algebraic Manipulation, E.W. Ng eds, Lecture Notes in Computer Science, LNCS 72, pp. 3–21. 1979.

[9] B. Buchberger. *Gröbner Bases : A Short Introduction for Systems Theorists.* In proceedings, EUROCAST 2001, 8th International Conference on Computer Aided System Theory - Formal Methods and Tools for Computer Science), R. Moreno-Diaz, B ; Buchberger, J.L. Freire eds, Lecture Notes in Computer Science, LNCS 2178. 2001.

[10] C. Carlet. *On cryptographic complexity of boolean functions.* In proceedings, 6th conference on Finite Fields with Applications to Coding Theory, Cryptography and Related Areas, G.L. Mullen, H.Stichtenoth, H. Tapia-Recillas eds, pp. 53–69. 2002.

[11] C. Carlet, P. Guillot. *Characterization of Binary Bent Functions.* Journal of Combinatorial Theory, Vol. A(76), pp. 328–335. 1996.

[12] F. Chabaud, S. Vaudenay. *Links between differential and linear cryptanalysis.* In proceedings, EUROCRYPT'94, Lecture Notes in Computer Science, LNCS 950, pp. 328–335. 1996.

[13] *CMU package : bddlib.*
`http://www.cs.cmu.edu/~modelcheck/bdd.html`

[14] *CUDD : CU Decision Diagram Package.*
`http://vlsi.colorado.edu/~fabio/CUDD/`

[15] N. Courtois. *The Security of Hidden Filed Equations (HFE).*

[16] W. Diffie, M.E. Hellman. *New directions in Cryptography.* IEEE Transactions on Information Theory, Vol. IT-22, pp. 644-654. 1976.

[17] V. Dubois, L. Granboulan, J. Stern. *An Efficient Provable Distinguisher for HFE.* In proceedings, ICALP'2006, Lecture Notes in Computer Science, LNCS 4052, pp. 56–167. 2006.

[18] V. Dubois, L. Granboulan, J. Stern. *Cryptanalysis of HFE with Internal Perturbation.* In proceedings, Public Key Cryptography 2007, Lecture Notes in Computer Science, LNCS 4450, pp. 249–265. 2007.

[19] J.H. Ellis. *The possibility of secure non secret encryption.* CESG Research Report, 1970.

[20] J.C. Faugère. *A new efficient algorithm for computing Gröbner bases (F4).* Journal of Pure and Applied Algerba, Vol. 139(1–3), pp. 61–88. 1999.

[21] J.C. Faugère. *A new efficient algorithm for computing Gröbner bases without reduction to zero F5.* In proceedings, International SYmposium on Symbolic and Algebraix Computation ISSAC'02, pp. 75–83, T. Mora eds, ACM Press, July 2002.

[22] J.C. Faugère, A. Joux. *Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases.* In proceedings, Advances in Cryptology - CRYPTO 2003, D. Boneh eds, Lecture Notes in Computer Science, LNCS 2729, pp. 44-60. 2003.

[23] E. Férard, F. Rodier. *Nonlinearity of some Boolean functions.* In proceedings, BFCA'06, J-F. Michon, P. Valarcher, J-B. Yunes eds, Publications des Universités de Rouen et du Havre.2006.

[24] C. Fontaine. *Contribution à la recherche de fonctions booléennes hautement non-linéraires et au marquage d'images en vue de le protection des droits d'auteur.* Thèse de doctorat. Université Paris 6. 1998.

[25] M.R. Garey, D.S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness.* W.H. Freeman Company. 1979.

[26] L. Granboulan, A. Joux, J. Stern. *Inverting HFE is Quasipolynomial.* In proceedings, CRYPTO'2006, Lecture Notes in Computer Science, LNCS 4117, pp. 345–356. 2006.

[27] C. Gröpl. *Binary Decision Diagrams for Random Boolean Functions.* PhD Thesis, Humboldt-Universität zu Berlin. 1999.

[28] C. Gröpl, H.J. Prömel, A. Srivastav. *Size and structure of random ordered binary decision diagrams (extended abstract).* In proceedings, STACS'98, Lecture Notes on Computer Science, LNCS 1373, pp. 238–248. 1998.

[29] C. Gröpl, H.J. Prömel, A. Srivastav. *On the evolution of the worst-case obdd size.* Information Processing Letters, Vol. 77, pp. 1–7. 2001.

[30] M. Heap, M.R. Mercer. *Least upper bounds on obdd size.* IEEE Trans. Computer, Vol. 43, pp. 764–767. 1994.

[31] X. Jiang, J. Ding, L. Hu. *Kipnis-Shamir's Attack on HFE Revisited.* 3rd International SKLOIS Conference on Information Security and Cryptology, Inscrypt 2007 (formerly CISC). August-September 2007, Xining, China. 2007. Lectures Notes in Computer Science, LNCS. 2007.

[32] D. Kahn. *The CODE-BREAKERS : The comprehensive history of secret communication from ancient times to the Internet.* Scribner. 1996.

[33] A. Kipnis, A. Shamir. *Cryptanalysis of the HFE public key cryptosystem by relinearization.* In proceedings, CRYPTO'99. Lectures Notes in Computer Science, LNCS 1666, pp. 19–30. 1999.

[34] D.E. Knuth. *The Art of Computer Programming.* Vol. 2. Addison-Wesley. 1998.

[35] N. Koblitz. *Algebraic Aspects of Cryptology.* In collection, Algorithms and Computation in Mathematics, Vol. 3. Springer. 1999.

[36] M. Krause. *BDD-based cryptanalysis of keystream generators.* In Proceedings, EUROCRYPT 2002. LNCS 2332, pp. 222—237. 2002.

[37] C.Y. Lee. *Representation of Switching Circuits by Binary-Decision Programs.* Bell Systems Technical Journal, Vol. 38, pp. 985–999. 1959.

[38] *Selected Area in Cryptography.* Proceedings, 8th Annual International Workshop, SAC 2001, Toronto, Canada, August 2001. S. Vaudenay,

A.M. Youssef eds. Lectures Notes in Computer Science, LNCS 2251. 2001.

[39] *Advances in Cryptology - EUROCRYPT 2002.* Proceedings, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April/May 2002. L. Knudsen eds. Lectures Notes in Computer Science, LNCS 2332. 2002.

[40] *Theory of Cryptography.* Proceedings, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, February 2004. M. Naor eds. Lectures Notes in Computer Science, LNCS 2951. 2004.

[41] K. Mahler. *p-adic numbers and their functions.* Cambridge Tracts in Mathematics, Cambridge University Press (2nd edition), 1981.

[42] T. Matsumoto, H. Imai. *Public quadratic polynomial-tuples for efficient signature-verification and message encryption.* In proceedings, Advances in Cryptology EUROCRYPT'88, LNCS 330, pp. 419–453. 1988.

[43] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. *Handbook of Applied Cryptography. 2nd edition.* CRC Press. 1997.

[44] J.-F. Michon, J.-M. Champarnaud. *Automata and Binary Decision Diagrams.* In proceedings, WIA'98, Workshop on Implementing Automata, J.-M. Champarnaud, D. Maurel, D. Ziadi eds, Lecture Notes in Computer Science, LNCS 1660, pp. 178–182. 1999.

[45] J.-F. Michon, P. Valarcher, J.-B. Yunès. *Malher's expansion and Boolean functions.* Journal of Integer Sequences. Vol. 10(3). 2007.

[46] J.-F. Michon, P. Valarcher, J.-B. Yunès. *Sequence of Enumeration of QROBDD.* On-Line Encyclopedia of Integer Sequences. Sequence A100344. 2004.

[47] J.-F. Michon, P. Valarcher, J.-B. Yunès. *On Maximal QROBDD's of Boolean Functions.* RAIRO ITA/TIA, Vol. 39(4), October-December 2005. Ref. ITA0442.

[48] J.-F. Michon, P. Valarcher, J.-B. Yunès. *HFE and BDDs : A Practical Attempt at Cryptanalysis.* In Proceedings, International Workshop on Coding, Cryptography and Combinatorics, CCC'03, China. K. Q. Feng, H. Niederreiter, C. P. Xing, eds. Progress in Computer Science and Applied Logic, Vol. 23, pp. 237–246. Birkhaüser, 2004. isbn 3-7643-2429-5.

[49] J.-F. Michon, J.-B. Yunès. *A HFE cryptosystem generator.* `http://www.liafa.jussieu.fr/~yunes/HFE/`

[50] S.I. Minato, N. Ishiura, S. Yajima. *Shared binary decision diagram with attributed edges for efficient boolean function manipulation.* In proceedings, 27th ACM/IEEE Design Automaton Conference, Orlando, FL, pp. 52–57. 1991.

[51] *NTL Library.* `http://www.shoup.net/`

[52] J. Patarin. *Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88.* In proceedings, Advances in Cryptology - CRYPTO'95. Lecture Notes in Computer Science, LNCS 963, pp. 248–261. 1995.

[53] J. Patarin. *Hidden Fields equations (HFE) and isomorphim of polynomials (IP) : two new families of asymetric algorithms.* In proceedings, EUROCRYPT'96. Lecture Notes in Computer Science, LNCS 1070, pp. 33–48. 1996.

[54] W. Paul. *A 2.5n lower bound on the combinatorial complexity of boolean functions.* SIAM J. Compu. 6, pp. 427–443. 1977.

[55] A. Robert. *A Course in p-adic Analysis.* GTM 198, Springer. 2000.

[56] F. Rodier. *On the nonlinearity of Boolean functions.* In proceedings, WCC 2003, Workshop on Coding and Cryptography 2003, D. Augot, P. Charpin, G. Kabatianski eds, INRIA, pp. 397–405. 2003.

[57] F. Rodier. *Asymptotic nonlinearity of Boolean functions.* Designs, Codes and Cryptography, 401. 2006.

[58] S. Singh. *The Code Book : The Secret History of Codes and Code Breaking.*

[59] N.J. Sloane. *The On-Line Encyclopedia of Integer Sequences.*

[60] J. Stern. *La science du secret.* Odile Jacob. 1998.

[61] I. Wegener. *Branching Programs and Binary Decision Diagrams : Theory and Applications* SIAM Monographs on Discrete Mathematics and Applications. 2000.

[62] I. Wegener. *The Complexity of Boolean Functions.* John Wiley and Sons. 1987.

[63] C. Wolf, P. Fitzpatrick, S.N. Foley, E. Popovici. *HFE in Java : Implementing Hidden Field Equations for Public Key Cryptography.* Irish Signals and Systems Conference. 2002.

[64] J. von zur Gathen, J. Gerhard. *Modern Computer Algebra. 2nd edition.* Cambridge University Press. 2003.

[65] M.J. Williamson. *Non-secret Encryption Using Finite Field.* 1974.

[66] J.-B. Yunès. *HFE experiments.*
`http://www.liafa.jussieu.fr/~yunes/HFE/`

[67] G. Brochier, C. Card, J. Vaughan, J.-B. Yunès. *Experiments on "Singh's Cipher Challenge".* Unpublished materials. 1999–2000.

[68] J.-B. Yunès. *A ROBDD package.*
`http://www.liafa.jussieu.fr/~yunes/HFE/`

Annexe A

Tables de transitions

•	$	•	A	B	C	D	E	F
$								
•	•	•					•	•
A	A	B		B			B	B
B	A	B	F	C			B	B
C		E	E	D				F
D			F	C				
E		E						F
F		•					•	•

A	$	•	A	B	C	D	E	F
$		A	G	A				
•		A					A	
A	G	A	G	A		A	A	A
B	A		A					A
C	A	A	A		A			
D	A		A		A			
E	A		A			A		
F	A		A	A				

B	$	•	A	B	C	D	E	F
$								
•							•	•
A		B	A	B	B			A
B		C		C	B	B	A	A
C		D		D	B	B		A
D		C		C	B	B		A
E								
F								

C	$	•	A	B	C	D	E	F
$								
•								
A		D					B	B
B		B	F	C	C	B		F
C		D	F	C	C	D		F
D			F	C	C			F
E								
F		E	F				•	•

D	$	•	A	B	C	D	E	F
$								
•								
A							•	B
B		C	E	D	C	D	F	F
C			E	D		D	F	F
D		C	E	D	C	D	F	F
E								
F								

E	$	•	A	B	C	D	E	F
$								
•							•	•
A		D					D	D
B		B	F				B	B
C							•	•
D		B	C				B	B
E		E					E	E
F		E					E	E

F	$	•	A	B	C	D	E	F
$								
•		F	F				F	F
A			A					
B		A	A		C		A	A
C		E	F				•	•
D					C			A
E		F					F	F
F		E	F				•	•

FIG. A.1 – La fonction de transition de la solution de Balzer.

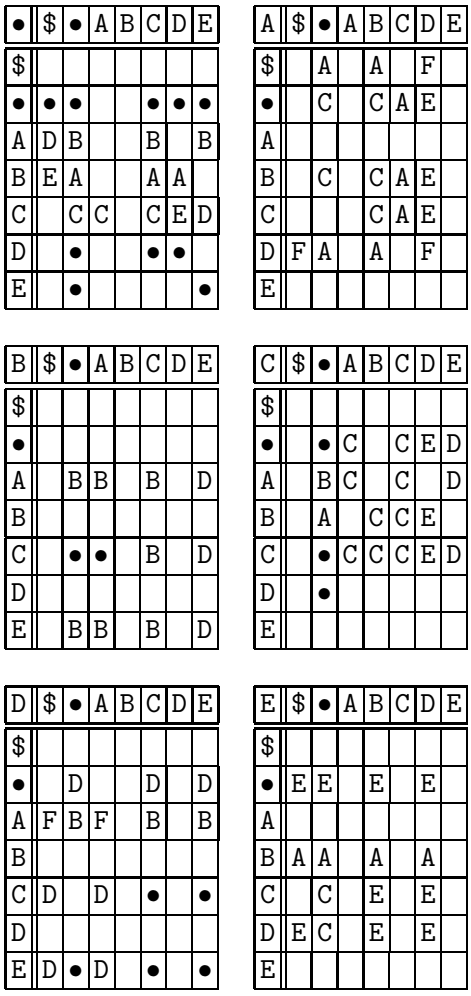
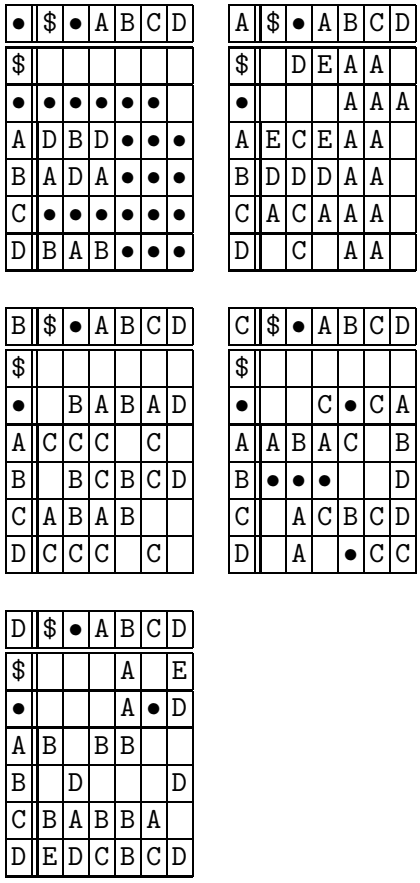


FIG. A.2 – La fonction de transition de la solution de Gerken.



•	$	•	A	B	C	D
$						
•	•	•	•	•	•	
A	D	B	D	•	•	•
B	A	D	A	•	•	•
C	•	•	•	•	•	•
D	B	A	B	•	•	•

A	$	•	A	B	C	D
$		D	E	A	A	
•				A	A	A
A	E	C	E	A	A	
B	D	D	D	A	A	
C	A	C	A	A	A	
D		C		A	A	

B	$	•	A	B	C	D
$						
•		B	A	B	A	D
A	C	C	C		C	
B		B	C	B	C	D
C	A	B	A	B		
D	C	C	C		C	

C	$	•	A	B	C	D
$						
•			C	•	C	A
A	A	B	A	C		B
B	•	•	•			D
C		A	C	B	C	D
D		A		•	C	C

D	$	•	A	B	C	D
$				A		E
•				A	•	D
A	B		B	B		
B		D				D
C	B	A	B	B	A	
D	E	D	C	B	C	D

FIG. A.3 – La fonction de transition de la solution de Mazoyer.

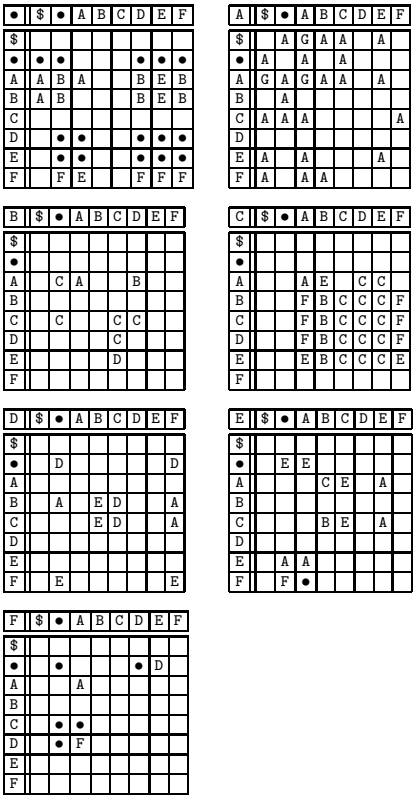
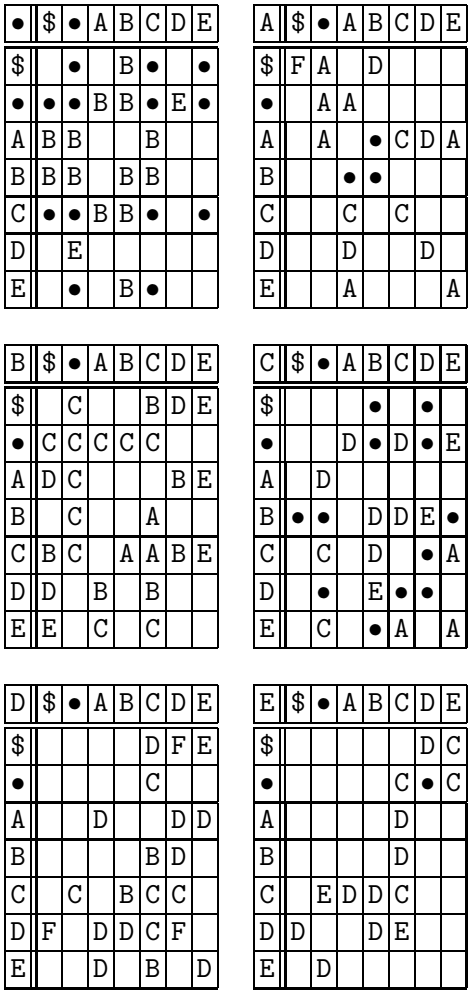


FIG. A.4 – La fonction de transition de la solution de Noguchi.



•	$	•	A	B	C	D	E
$		•		B	•		•
•	•	•	B	B	•	E	•
A	B	B			B		
B	B	B		B	B		
C	•	•	B	B	•		•
D		E					
E		•		B	•		

A	$	•	A	B	C	D	E
$	F	A		D			
•		A	A				
A		A		•	C	D	A
B			•	•			
C			C		C		
D			D			D	
E			A				A

B	$	•	A	B	C	D	E
$		C			B	D	E
•	C	C	C	C	C		
A	D	C				B	E
B		C			A		
C	B	C		A	A	B	E
D	D		B		B		
E	E		C		C		

C	$	•	A	B	C	D	E
$				•		•	
•			D	•	D	•	E
A		D					
B	•	•		D	D	E	•
C		C		D		•	A
D		•		E	•	•	
E		C		•	A		A

D	$	•	A	B	C	D	E
$					D	F	E
•					C		
A			D			D	D
B					B	D	
C		C		B	C	C	
D	F		D	D	C	F	
E			D		B		D

E	$	•	A	B	C	D	E
$						D	C
•					C	•	C
A					D		
B					D		
C		E	D	D	C		
D	D			D	E		
E		D					

FIG. A.5 – La fonction de transition de la solution de Yunes [108].

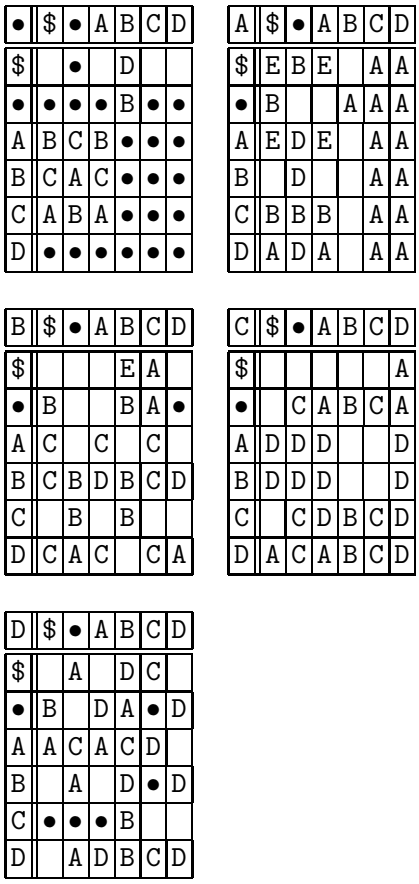


FIG. A.6 – La fonction de transition de la solution de Settle.

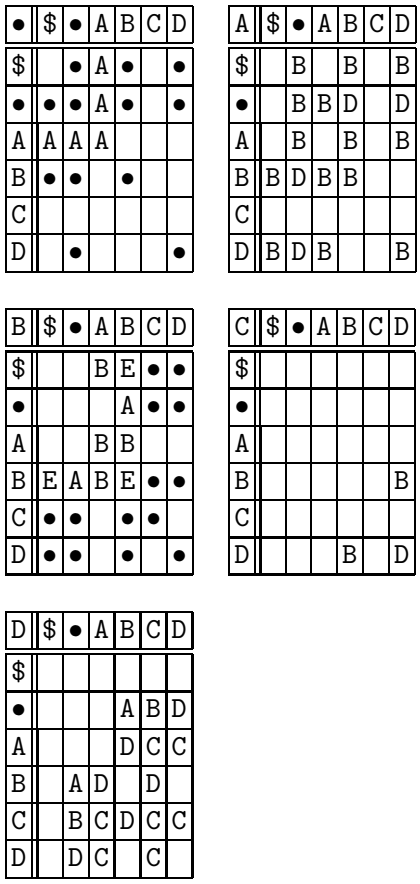


FIG. A.7 – La fonction de transition de la solution de Umeo.

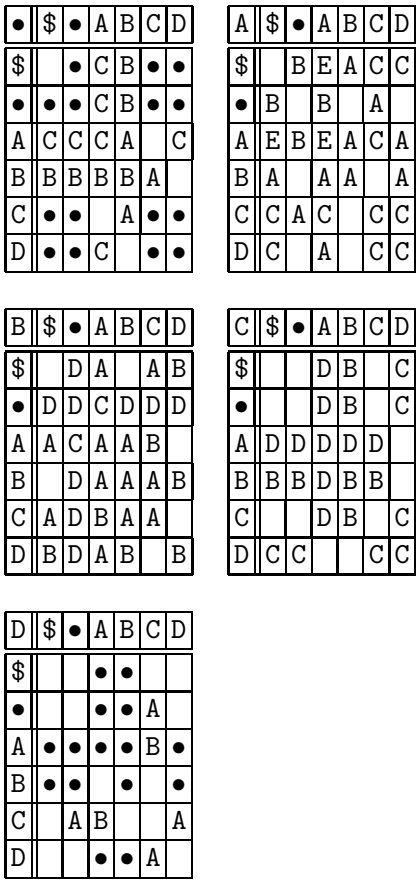


FIG. A.8 – La fonction de transition de la solution de Yunès à 6 états.

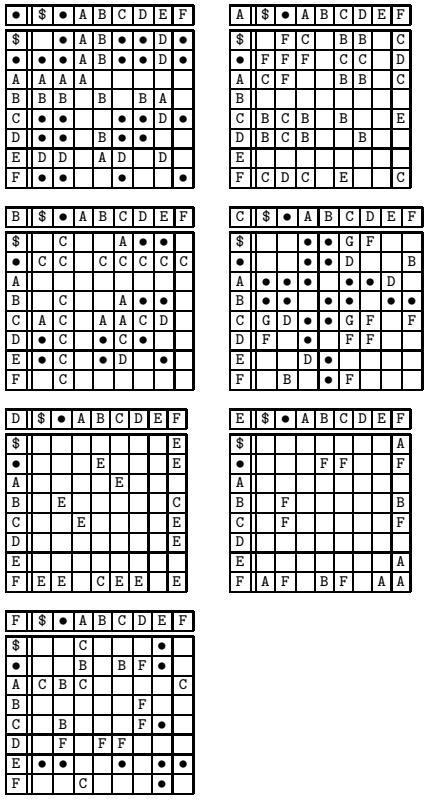


FIG. A.9 – La fonction de transition de la solution de Yunès en temps $4n$.

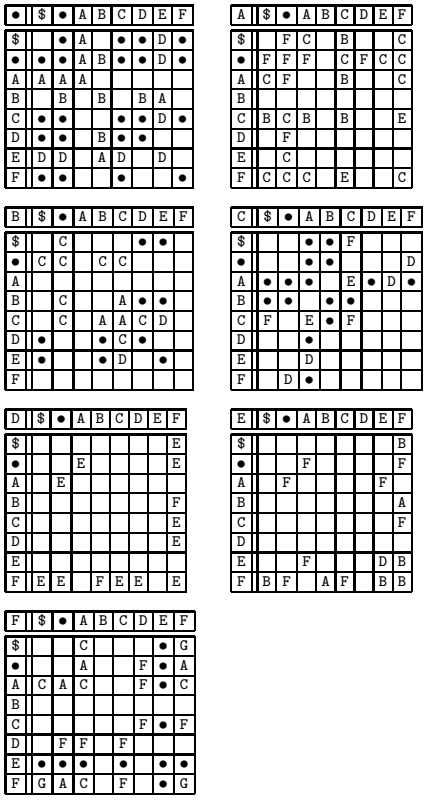


FIG. A.10 – La fonction de transition de la solution de Yunès en temps $5n$.

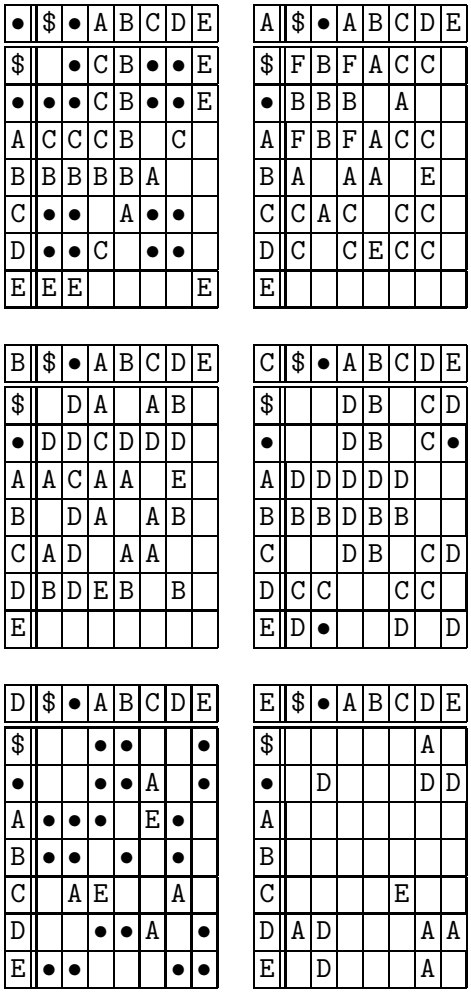


FIG. A.11 – La fonction de transition de la solution de Yunès en temps $4n$ (stratégie alternative).

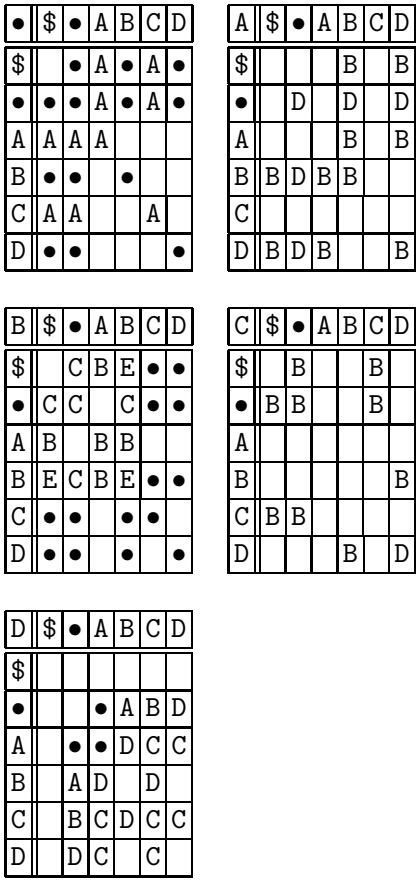


FIG. A.12 – La fonction de transition de la solution de Yunès en temps $4n$.

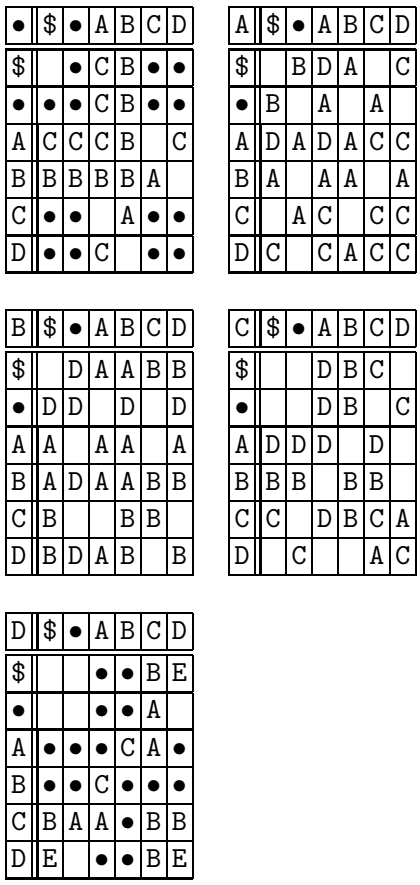


FIG. A.13 – La fonction de transition de la solution de Yunès en temps $6n$.

Annexe B

Liste des travaux

Thèse

[1] J.-B. Yunès, *Synchronisation et Automates Cellulaires : la ligne de fusiliers.* (PhD thesis) Thèse LITP 93/01, 1993.

Publications dans des revues/journaux internationaux avec comité

[1] J.-B. Yunès. *Seven States Solutions to the Firing Squad Synchronization Problem.* Theoretical Computer Science, 127(2):313-332, 1994.

[2] J.-F. Michon, P. Valarcher, J.-B. Yunès. *On Maximal QROBDD's of Boolean Functions.* RAIRO ITA/TIA, Vol. 39(4), ITA0442, pp. 677–686, October-December 2005.

[3] J.-B. Yunès. *Fault Tolerant Solutions to the Firing Squad Synchronization Problem in Linear Cellular Automata.* Journal of Cellular Automata 1(3):253–268, 2006.

[4] J.-F. Michon, P. Valarcher, J.-B. Yunès. *Malher's expansion and Boolean functions.* Journal of Integer Sequences. Vol. 10(3). 2007.

[5] J.-B. Yunès. *An Intrinsically non Minimal-Time Minsky-like 6-States Solutions to the Firing Squad Synchronization Problem.* To appear in RAIRO ITA/TIA. 2008.

Publications dans des actes de congrès internationaux avec comité

[1] J.-F. Michon, P. Valarcher, J.-B. Yunès. *HFE and BDDs : A Practical Attempt at Cryptanalysis.* In Proceedings, International Workshop

on Coding, Cryptography and Combinatorics, CCC'03, China. K. Q. Feng, H. Niederreiter, C. P. Xing, eds. Progress in Computer Science and Applied Logic, Vol. 23, pp. 237–246. Birkhaüser, 2004. isbn 3-7643-2429-5.

[2] J.-B. Yunès. *Simple new algorithms which solve the FSSP : A 7-states 4n-steps solution.* In proceedings, J. Durand-Lose, M. Margenstern eds, 5th International Conference on Machines, Computations and Universality, MCU 2007, Lecture Notes on Computer Science, LNCS 4664, pp. 316–324. 2007.

Communications dans des congrès internationaux avec comité

[1] J.-B. Yunès. *Fault Tolerant Solutions to the Firing Squad Synchronization Problem.* Dagstuhl seminar 9510, IFIP TC1 WG1.5 , Germany. 1995.

[2] J.-F. Michon, P. Valarcher, J.-B. Yunès. *HFE and BDD : A practical attempt at cryptanalysis.* International Workshop on Coding, Cryptography and Combinatorics. China. Cancelled by SARS syndrom. June, 2003.

[3] J.-B. Yunès. *Simple new algorithms which solve the FSSP. (Updated contents)* 5th International Conference on Machines, Computations and Universality, MCU 2007, Orléans, France, September 10–14, 2007.

Communications dans des congrès internationaux sans comité

[1] J.-B. Yunès. *An Implementation of a fault-tolerant scheme of the FSSP.* International Workshop on Cellular Automata, AUTOMATA 1999. IFIP TC1 WG1.5. ENS, Lyon France. 1999.

[2] J.-B. Yunès. *Revisiting existing solutions to the firing squad synchronization problem.* Workshop on Symbolic Dynamic and Coding. Marne-la-vallée, France, July 2–4, 2007.

[3] J.-B. Yunès. *New extensions to some firing squad synchronization solutions.* 13th International Workshop on Cellular Automata, AUTOMATA 2007. IFIP TC1 WG1.5. Fields Institute, Toronto Canada. August 27–29, 2007.

Communications dans des congrès nationaux sans comité

[1] J.-B. Yunès. *Automates cellulaires et Théorie des nombres.* Journées Arithmétiques Faibles, Paris, 1992.

[2] J.-B. Yunès. *Solutions tolérantes aux pannes pour la synchronisation d'une ligne de fusiliers.* ASMICS Workshop, Roanne, 1995.

Articles soumis

[1] J.-B. Yunès. *Cellular Automata Synchronizers Insensitive to some Initial Conditions.* Submited to Theoretical Computer Science. 2007.

[2] J.-B. Yunès. *A 4-states algebraic solution to linear cellular automata synchronization.* Submitted to Information Processing Letters. 2007.

[3] J.-B. Yunès. *Known CA synchronizers made insensitive to the initial state of the initiator.* Submitted to Journal of Cellular Automata. 2007.

Articles en préparation

[1] J.-B. Yunès. *About linear-time synchronization of cellular automata.* In preparation to answer a Special Call for Papers in FI issued after MCU'07. 2008.

[2] J.-B. Yunès. *A simple Goto-like solution to the FSSP.* In preparation. 2007.

Travaux Divers

[1] J.-F. Michon, P. Valarcher, J.-B. Yunès. *Sequence of Enumeration of QROBDD.* On-Line Encyclopedia of Integer Sequences. Sequence A100344. 2004.

[2] J.-B. Yunès. *HFE experiments.* `http://www.liafa.jussieu.fr/~yunes/HFE/`

[3] G. Brochier, C. Card, J. Vaughan, J.-B. Yunès. *Experiments on "Singh's Cipher Challenge".* Unpublished materials. 1999–2000.

Réalisations logicielles

[1] J.-F. Michon, J.-B. Yunès. *A HFE cryptosystem generator.* `http://www.liafa.jussieu.fr/~yunes/HFE/`

[2] J.-B. Yunès. *CA - LFSSP Explorer.* `MacOSX`® version. `http://www.liafa.jussieu.fr/~yunes/ca/`

[3] J.-B. Yunès. *A ROBDD package.* `http://www.liafa.jussieu.fr/~yunes/HFE/`

Édition

[1] Proceedings of MCU/UMC'95. 1st international workshop on Universal Machines and Computation. M. Margenstern, J.-B. Yunès. Rapport Technique LITP/IBP. 1995.

[2] Proceedings of BFCA'05, 1st International Workshop on Boolean Functions : Cryptography and Applications, March 2005, Rouen, France. J.-F. Michon, P. Valarcher, J.-B. Yunès eds. Presses Universitaires de Rouen et du Havre. 2005.

[3] Proceedings of BFCA'06, 2nd International Workshop on Boolean Functions : Cryptography and Applications, March 2006, Rouen, France. J.-F. Michon, P. Valarcher, J.-B. Yunès eds. Presses Universitaires de Rouen et du Havre. 2006.

[4] Proceedings of BFCA'07, 3rd International Workshop on Boolean Functions : Cryptography and Applications, May 2007, Paris, France. J.-F. Michon, P. Valarcher, J.-B. Yunès eds. Presses Universitaires de Paris 7. 2007. À paraître.

Livres

[1] J.-M. Rifflet, J.-B. Yunès. *UNIX : programmation et communication.* Dunod. 2003.

Traductions

[1] C. Liu, J. Peek, R. Jones, B. Buus, A. Nye. *Systèmes d'information sur Internet : Installation et mise en œuvre.* O'Reilly International Thomson. 1996.

[2] D. Cameron, B. Rosenblatt. *Introduction à GNU Emacs.* O'Reilly. 1997.