

République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la recherche scientifique



Université Mentouri de Constantine
Faculté des sciences de l'ingénieur

Département d'informatique

N° d'ordre : 101 / Mag / 2009

N° série : 002 / inf / 2009

Mémoire

Pour l'Obtention du diplôme de Magistère en Informatique

Thème :

Evolution d'automates cellulaires par algorithmes génétiques quantiques sur un environnement parallèle

Soutenu le : 28 / 04 / 2009

Présenté par :

LABOUDI Zakaria

Dirigé par :

Dr. CHIKHI Salim

Devant le jury composé de :

Prof. M.Benmohammed

Dr. S. Chikhi

Dr. F. Hachouf

Dr. D-E Saidouni

Université de Constantine

Université de Constantine

Université de Constantine

Université de Constantine

Président

Rapporteur

Examineur

Examineur

Année Universitaire 2008-2009

Dédicace

A toute ma famille

Remerciements

« Les aigles volent souvent aussi bas que les poules, mais les poules ne volent jamais aussi haut que des aigles ».

Etant des poussins qui se nourrissent de sciences et de culture, nous voilà pourvus d'ailes plumées et gagnant les airs ouvrants la porte vers le monde informatique.

Tout d'abord, je remercie DIEU, le tout puissant, qui m'a donné la force, la patience et la volonté pour accomplir ce modeste travail.

Tout mon respect et ma gratitude à mon cher honorable encadreur **Dr. CHIKHI Salim** qui m'a été l'enseignant, le père et le conseiller et qui m'a soutenu tout le long de ma recherche.

Je tiens à présenter de tout mon cœur mes remerciements et ma reconnaissance très chaleureuse au Professeur M. BENMOHAMMED pour avoir accepté de présider mon jury de soutenance.

Je tiens à remercier vivement de tout mon cœur Dr. F. HACHOUF ainsi que Dr. D-E. SAIDOUNI, pour avoir bien voulu juger ce travail et d'avoir pris part à mon jury.

Salutation respectueuse pour tout enseignant qui a contribué à l'obtention du titre Magistère en informatique.

Je tiens à remercier vivement, Mr. RIGHI Hamza qui m'a vraiment aidé lorsque j'avais besoin d'une exécution parallèle de mon application, je lui suis donc très reconnaissant d'avoir bien voulu m'aider pour accomplir ce modeste travail.

Ainsi que tous qui ont été de près ou loin de précieuse aide.

A tous ceux que j'aime et que je respecte.

Les derniers remerciements vont à ma famille pour tout ce qu'elle a fait pour moi et sans laquelle rien de ce qui est entre vos mains aujourd'hui n'aurait été réalisé.

A tous je dis merci

عرفت دراسة الأنظمة المعقدة اهتماما كبيرا من طرف الباحثين في مجال علوم التعقيد و كذلك الإعلام الآلي. تعتمد دراسة مثل هذه الأنظمة على تمثيلها عن طريق نماذج و من ثم استخراج الخصائص المرغوب في دراستها. تعتبر الآلات الخلوية أداة قيمة لدراسة مثل هذه الأنظمة لأنها يمكن أن تسفر عن سلوك جد معقد انطلاقا من مجموعة قواعد يمكن أن تكون بسيطة جدا. ومع ذلك ، فإن صعوبة استغلال إمكاناتها على نحو فعال غالبا ما حد من استخدامها في معالجة العديد من الإشكاليات. لذلك فكر العلماء في إيجاد طريقة آلية لتصميمها عن طريق الخوارزميات التطورية و التي بإمكانها استكشاف عالم الآلات الخلوية و نتكلم إذا عن الآلات الخلوية التطورية. مهمتنا من خلال هذا البحث هي أساسا القيام بوضع نهج يتسم بفعالية وكفاءة لتصميم الآلات الخلوية. النهج المقترح في هذا البحث (الآلات الخلوية التطورية الكمية) يعتمد أساسا على استعمال الخوارزميات الجينية الكمية من أجل تصميم آلات خلوية و هذا في نظام يعمل بمبدأ التوازي. النتائج التجريبية أظهرت تحسنا ملحوظا مقارنة بالنموذج الكلاسيكي المستعمل في تصميم الآلات الخلوية من خلال أداء عالي التوازي مع الحفاظ على الكفاءة.

كلمات مفتاحية : نظام معقد، آلة خلوية ، الخوارزميات التطورية ، إعلام آلي كمي.

Résumé

L'étude des systèmes complexes a récemment connu un intérêt croissant par des chercheurs en sciences de la complexité et de l'informatique. Leur étude passe généralement par leur modélisation. Il semble que la classe du modèle abstrait d'automates cellulaires constitue un outil précieux pour l'étude de tels systèmes dans la mesure où ils peuvent exhiber un comportement global énorme partant seulement d'un ensemble de règles pouvant être très simples. Cependant, la difficulté d'exploiter efficacement leur potentiel a souvent limité leur utilisation. Pour surmonter ce problème, les chercheurs ont pensé d'automatiser leur processus de conception en faisant appel aux algorithmes évolutionnaires qui peuvent explorer des espaces de recherche des automates cellulaires et on parle alors des automates cellulaires évolutionnaires.

Notre travail consiste essentiellement à développer une approche qui soit à la fois efficace et performante pour la conception des automates cellulaires. L'approche proposée (automate cellulaire évolutionnaire quantique) repose sur l'évolution des automates cellulaires par des algorithmes génétiques quantiques dans un environnement parallèle. Les résultats expérimentaux ont montré une nette amélioration par rapport au modèle classique des automates cellulaires évolutionnaires dans la mesure où ils s'exécutent dans des environnements parallèles en haute performance tout en préservant l'aspect efficacité.

Mots-clés : Système Complexe, Automate Cellulaire, Algorithme Evolutionnaire, Informatique Quantique.

Abstract

The study of complex systems has recently known a growing interest by researchers in sciences of complexity and information technology. Their study is usually done by modeling. It seems that the class of the abstract model of cellular automata is a valuable tool for studying such systems as they can produce a huge global behavior from only a set of rules that can be very simple. However, the difficulty of effectively exploit their potential is often limited their use. To overcome this problem, researchers have thought to automate their design process using evolutionary algorithms that can explore research spaces of cellular automata; this technique is called evolutionary cellular automata.

Our research work is basically to develop an approach that is both effective and efficient for designing cellular automata. The proposed approach (quantum evolutionary cellular automata) is based on evolution of cellular automata by quantum genetic algorithms in a parallel environment. The experimental results have shown a marked improvement over the classical model of evolutionary cellular automata as they can be run in parallel environments with high performance while preserving their efficiency.

Keywords: Complex System, Cellular Automata, Evolutionary Algorithm, Quantum Computing.

Table des matières

Introduction générale	1
------------------------------------	---

Chapitre I

Les automates cellulaires évolutionnaires : Etat de l'art

1. Introduction	6
2. La naissance des automates cellulaires évolutionnaires	7
2.1. La géométrie de l'espace des AC	9
2.2. Hypothèse de Langton (Le paramètre λ)	11
2.3. L'idée de départ de Packard	14
2.4. Le problème de la classification de densité	16
2.5. L'institut de Santa Fe (SFI) réexamine les résultats de Packard	19
3. La recherche autour du problème $\rho_c = 1/2$	21
4. La généralisation pour résoudre d'autres problèmes	31
5. Conclusion	36

Chapitre II

Les automates cellulaires évolutionnaires : Etudes techniques et formelles

Introduction	39
Partie I : Automates cellulaires.....	40
1. Origines et développements.....	40
2. Définitions de base	42
3. Terminologie	43
4. Universalités des AC	44
4.1. Universalité Turing	45
4.1.1. Machine de Turing vs automate cellulaire.....	45
4.1.2. L'universalité Turing des AC.....	46
4.2. L'universalité intrinsèque des AC.....	47
5. Le phénomène d'émergence dans l'univers des AC.....	48
Partie II : Algorithmes génétiques	51
1. Source d'inspiration.....	51
2. Algorithmes génétiques	52

3. Terminologie	53
4. Comment ça fonctionne ?.....	55
4.1. Codage des chromosomes	56
4.2. Initialisation de la population	57
4.3. Evaluation des individus.....	58
4.4. Sélection	58
4.5. Croisement.....	60
4.6. Mutation.....	62
4.7. Remplacement	63
Partie III : Vers des automates cellulaires évolutionnaires.....	64
1. Automates cellulaires évolutionnaires.....	64
2. D'un automate cellulaire vers un chromosome.....	64
3. Construction d'un automate cellulaire évolutionnaire	65
3.1. Initialisation de la population	65
3.2. Evaluation des individus.....	66
3.3. Opérations génétiques.....	66
4. Etude de cas	66
4.1. Codage des chromosomes	67
4.2. Initialisation de la population	67
4.3. Evaluation des individus.....	68
4.4. Opérations génétiques.....	68
Conclusion	69

Chapitre III

Vers des automates cellulaires évolutionnaires quantiques

1. Introduction	71
2. Les ordinateurs quantiques entre la réalité et l'espoir.....	72
3. Introduction à l'informatique quantique	75
3.1. Source d'inspiration	75
3.2. Notions de base.....	76
3.2.1. Les qubits.....	76
3.2.2. Principes des calculateurs quantiques.....	77
3.2.3. Réalisation d'un ordinateur quantique (portes quantiques).....	78

3.3. Terminologie	80
4. Algorithmes génétiques quantiques (AGQ)	81
4.1. Codage des chromosomes quantiques	82
4.2. Initialisation de la population	82
4.3. Evaluation des individus.....	82
4.4. Opérations génétiques quantiques	82
5. Extension pour évoluer des AC	85
6. Performances des AGQ	86
6.1. La présence du parallélisme	86
6.2. Application des conditions de Bernstein sur les AGQ	86
6.2.1. L'évaluation	87
6.2.2. La sélection	87
6.2.3. Le croisement	87
6.2.4. La mutation.....	88
6.2.5. L'interférence	88
6.2.6. Le remplacement	89
6.3. Les limites du parallélisme	89
6.3.1. La loi d'Amdahl	89
6.3.2. Interprétation	89
6.3.3. La loi de Gustavson	91
7. Algorithme proposé basé sur les AGQ pour l'évolution des AC.....	91
8. Conclusion	94

Chapitre IV

Parallélisation des automates cellulaires évolutionnaires quantiques

1. Introduction	96
2. Plateforme pour la simulation des AC.....	97
2.1. Idée de départ.....	97
2.2. Transformation des configurations	98
2.3. Architecture de la plateforme	99
2.3.1. La couche basse (Low Level).....	100
2.3.2. La couche médiane (Medium Level).....	100
2.3.3. La couche haute (Hight Level)	102

3. Adaptation des ACEVQ pour supporter un traitement parallèle	104
3.1. Parallélisation de l'algorithme proposé	104
3.2. La mise en œuvre d'un ACEVQ parallèle	105
3.3. Implémentation de notre algorithme parallèle	109
3.3.1. Implémentation du site maitre	109
3.3.2. Implémentation des sites esclaves	115
4. Application des ACEVQ sur divers problèmes	116
4.1. Problème de la classification de densité	116
4.1.1. Les AC étudiés.....	116
4.1.2. Codage des fonctions de transitions	116
4.1.3. Initialisation de la population	117
4.1.4. Evaluation des individus.....	117
4.1.5. Interférence	118
4.2. Problème des bitmaps	118
4.2.1. Les AC étudiés.....	119
4.2.2. Codage des fonctions de transitions	120
4.2.3. Initialisation de la population	121
4.2.4. Evaluation des individus.....	121
4.2.5. Interférence	121
5. Résultats expérimentaux.....	122
5.1. Mesure d'efficacité des ACEVQ	122
5.1.1. Problème de la classification de densité	122
5.1.2. Problème des bitmaps	122
5.2. Mesure de performance des ACEVQ (séquentiel vs parallèle)	123
5.2.1. Durée moyenne d'une génération.....	124
5.2.2. Durée moyenne de communication	125
5.2.3. Accélération atteignable	126
5.3. Discussion.....	127
6. Conclusion	129
Conclusion générale	130
Bibliographie & Webographie	132
Annexe A	138
Annexe B	139

Liste des figures

1.1. Structure d'un espace d'AC en fonction de λ	12
1.2. Variations de la complexité en fonction de λ	12
2.1. Structure stable	49
2.2. Oscillateur clignotant.....	49
2.3. Planeur de cinq cellules	50
2.4. Principe de fonctionnement d'un AG	55
2.5. Structure d'un chromosome en codage binaire	56
2.6. Structure d'un chromosome en codage en nombres réels	57
2.7. Structure d'un chromosome en codage à base n ($n = 5$)	57
2.8. Tirage par roulette	59
2.9. Croisement en un point.....	61
2.10. Croisement en deux points	61
2.11. Opérateur de mutation	62
2.12. Calcul des locus des gènes.....	67
2.13. Structure d'un chromosome [512 bits].....	67
3.1. Développement des circuits selon la loi de Moore.....	74
3.2. La porte quantique de Hadamard.....	78
3.3. La porte quantique CNOT	79
3.4. La porte quantique de Toffoli	79
3.5. Exemple d'un circuit quantique	79
3.6. Principe de fonctionnement d'un AGQ	81
3.7. Structure d'un chromosome quantique.....	82
3.8. Mesure d'un chromosome quantique	83
3.9. Croisement quantique en un point	84
3.10. Mutation quantique.....	85
3.11. Principe de fonctionnement de l'AGQ de [31].....	92
4.1. Système de transformation des configurations.....	99
4.2. Architecture d'une plateforme pour des AC à N dimensions	99
4.3. Modèle Maître / Esclave.....	105
4.4. Structure des tâches parallèles	106
4.5. Nouvelle structure des tâches parallèles	108

4.6. Structure du site Maitre	109
4.7. Déroulement d'exécution des processus traitants.....	114
4.8. Structure d'un chromosome quantique [128 qubits]	117
4.9. Structure du chromosome quantique mesuré résultant [128 bits]	117
4.10. Ensembles de configurations de taille $100 * 100$	119
4.11. Quatre configurations du voisinage symétriquement équivalentes	120
4.12. Structure d'un chromosome quantique [102 qubits]	120
4.13. Structure du chromosome quantique mesuré résultant [102 bits]	120
4.14. Variations du temps moyen d'une génération en fonction du nombre de sites utilisés	125
4.15. Variations du temps moyen de communication en fonction du nombre de sites utilisés..	126
4.16. Variations de l'accélération en fonction du nombre de sites utilisés	127
A.1. Stratégies calculatoires découvertes à partir de l'étude du problème $\rho c = \frac{1}{2}$	138
B.1. AC optimal retenu par l'ACEVQ	139
B.2. Configurations résultantes du problème des bitmaps par l'AC optimal	139

Liste des tables

2.1. Table de marche de la machine de Turing étudiée	45
3.1. Bit vs Qubit.....	76
3.2. Table de vérité de la porte CNOT	78
3.3. Implémentation de la fonction mesure	83
3.4. Table de recherche pour la rotation des portes quantiques.....	93
4.1. Pseudo code du site maitre	110
4.2. Pseudo code du processus Dispatcher	110
4.3. Pseudo code de chaque processus traitant i	111
4.4. Ensemble de fonctions utilisées par chaque processus traitant i	112
4.5. Pseudo code de chaque site esclave i	115
4.6. Table de recherche pour la rotation des portes quantiques [problème $\rho c=1/2$].....	118
4.7. Table de recherche pour la rotation des portes quantiques [problème des bitmaps]...	121
4.8. Table des fitness du problème de la classification de densité	122
4.9. Table des fitness du problème des bitmaps	122
4.10. Tailles des populations manipulées	123
4.11. Temps moyens d'une génération.....	124
4.12. Temps moyens de la communication	125
4.13. Accélération atteignable	126

Liste des formules

2.1. Règle de mise à jour d'un AC	43
2.2. Cardinal de l'ensemble des fonctions de transitions	65
3.1. Notation de Dirac.....	76
3.2. Application des conditions de Bernstein sur les tâches d'un AGQ.....	87
3.3. Décomposition de la tâche d'évaluation.....	87
3.4. Décomposition de la tâche de reproduction	88
3.5. Décomposition de la tâche de mutation.....	88
3.6. Décomposition de la tâche d'interférence	89
3.7. La loi d'Amdahl.....	89
3.8. Interprétation de la loi d'Amdahl	89
3.9. Réécriture de la loi d'Amdahl	90
3.10. Application de la loi d'Amdahl sur un AGQ.....	90
3.11. Politique de rotation des qubits	93
4.1. Calcul du voisinage de Moore [AC à 2-D].....	98
4.2. Calcul du voisinage de Neumann [AC à 2-D].....	98
4.3. Calcul du voisinage de Moore [AC à 1-D].....	99
4.4. Calcul du voisinage de Neumann [AC à 1-D].....	99
4.5. Calcul des indexes des configurations virtuelles.....	99

Liste des acronymes

Acronyme	Description
AC	<u>A</u> utomate <u>C</u> ellulaire.
ACEV	<u>A</u> utomate <u>C</u> ellulaire <u>E</u> volutionnaire.
ACEVQ	<u>A</u> utomate <u>C</u> ellulaire <u>E</u> volutionnaire <u>Q</u> uantique.
ACEVQP	<u>A</u> utomate <u>C</u> ellulaire <u>E</u> volutionnaire <u>Q</u> uantique <u>P</u> arallèle.
AE	<u>A</u> lgorithme <u>E</u> volutionnaire.
AG	<u>A</u> lgorithme <u>G</u> énétique.
AGQ	<u>A</u> lgorithme <u>G</u> énétique <u>Q</u> uantique.
API	<u>A</u> pplication <u>P</u> rogramming <u>I</u> nterface
EPFL	<u>E</u> cole <u>P</u> olytechnique <u>F</u> édérale de <u>L</u> ausanne
GKL	<u>G</u> acs, <u>K</u> urdyumov et <u>L</u> evin
Ma TRD	<u>M</u> a <u>T</u> Ra <u>D</u> uction.
SFI	<u>S</u> anta <u>F</u> e <u>I</u> nstitute.

Introduction générale

Contexte

Un système complexe est tout d'abord un système qui comporte plusieurs composants indépendants arrangés selon une certaine structure permettant de décrire le système à plusieurs échelles. Les composants interagissent localement pour produire un comportement global bien défini et indépendant de leur structure interne. Cette classe de systèmes recouvre aussi bien les systèmes naturels de la cellule jusqu'à l'écosphère. L'une des propriétés intéressantes que présente un système complexe est celle de l'émergence : le niveau microscopique sous-jacent fait émerger des formes et des structures organisées au niveau macroscopique, ne pouvant que laisser admiratif. Comment de tels systèmes effectuent des calculs émergents partant seulement d'un ensemble d'interactions locales ?

La compréhension des systèmes complexes passe généralement par leur modélisation. Les modèles adoptés doivent fournir une reconstruction des données observées de manière expressive. Néanmoins, cela peut poser quelques problèmes difficiles, dits *problèmes inverses* : étant donné un corpus phénoménologique, comment choisir de modéliser les entités et les interactions pour garantir une meilleure compatibilité avec le corpus en question ? Comment trouver parmi les actions possibles, celles conduisant à des conséquences souhaitables ? D'autres questions peuvent également être posées.

Idéalement, le modèle doit être le plus simple possible mais aussi le plus représentatif. Il semble que la classe du modèle abstrait d'automates cellulaires possède ces caractéristiques du fait qu'elle constitue un outil précieux d'analyse de l'émergence dans la mesure où **de simples interactions locales peuvent engendrer un comportement global énorme**. On peut donc considérer les AC comme un outil très puissant pour la modélisation des systèmes complexes :

« L'étude des automates cellulaires peut être ramenée dans un cadre plus vaste, celui des sciences dites de la complexité, qui incluent l'étude des systèmes dynamiques chaotiques, la théorie de la morphogenèse, la vie artificielle, etc. Chacun de ces domaines possède ses propres paradigmes et les connexions entre eux restent très faibles. Nous pouvons dire que le seul lien fondamental entre les

disciplines des sciences de la complexité est d'utiliser l'ordinateur comme outil fondamental. » [32].

Problématique

Les automates cellulaires sont des systèmes dynamiques dont la caractéristique principale est de dépendre des règles pouvant être très simples, mais dont le comportement global résultant peut être très complexe. Ils constituent un nouveau paradigme caractérisé principalement par un traitement des problèmes selon une approche ascendante (du simple vers le complexe), parallèle et en déterminant les comportements des entités élémentaires de façon locale.

Cependant, la difficulté de concevoir un automate pour exhiber un comportement spécifique ou effectuer une tâche particulière a souvent limité leurs applications et donc empêché la maîtrise du comportement émergent. Ce problème est connu sous le nom de *problème inverse* où il est demandé de trouver un automate cellulaire (règles locales représentant des interactions entre des entités d'un niveau microscopique) possédant des propriétés globales présélectionnées (comportement observé au niveau macroscopique) [33].

Une solution possible est l'automatisation du processus de leur conception (programmation) via des algorithmes évolutionnaires pour renforcer leur viabilité du fait que les règles de transitions devaient être soigneusement produites à la main. Ceci peut conduire à trouver des moyens nécessaires pour l'ingénierie des calculs émergents sophistiqués dans les systèmes décentralisés et même résoudre d'autres problèmes calculatoires [21].

Dans la littérature, la plupart des travaux traitant l'apprentissage automatique des automates cellulaires mettent l'accent sur une seule technique : évoluer des automates cellulaires par des algorithmes évolutionnaires ce qui est connu sous le nom d'*automate cellulaire évolutionnaire*. Dans ce mémoire, nous tenterons de développer ce concept que nous baptisons *automate cellulaire évolutionnaire quantique*. Et ce, en faisant sortir quelques aptitudes calculatoires des automates cellulaires basées sur leur évolution par des algorithmes génétiques quantiques.

La difficulté principale dans la constitution d'une recherche sur ce sujet est l'absence d'ouvrages en la matière. La seule source restante est, heureusement, l'Internet. Même avec ce dernier, on est face à plusieurs problèmes sérieux : le volume des références est assez important ce qui ne facilite pas la tâche de recherche et les articles intéressants se trouvent

généralement disséminés dans de nombreuses revues et actes de conférences le plus souvent accessibles via un service payant. Nous avons donc conscience que ce travail restera partiel, tronqué et non exhaustif. Néanmoins nous avons déployé tous les efforts pour l'approcher dans plusieurs aspects et dont nous synthétiserons les informations les plus pertinentes.

Contributions

Nos contributions principales sur le sujet proposé peuvent se résumer en :

- La construction d'une référence solide sur les automates cellulaires évolutionnaires, concrétisée sous la forme d'un historique racontant leur évolution depuis la naissance jusqu'à nos jours. Ceci rend leur étude, par des étudiants ou par des chercheurs, plus facile, claire et accessible pour d'éventuelles contributions.
- La création d'une plateforme pour la simulation des automates cellulaires qui a la particularité de supporter théoriquement n'importe quel type d'automate cellulaire ($1-D$, $2-D$, $3-D$, ... $N-D$). Elle est également dotée de plus d'une technique pour l'évolution des automates cellulaires. Sa réalisation a nécessité un grand effort et une grande maîtrise du langage Java.
- Extension du concept d'automate cellulaire évolutionnaire en un autre appelé **automate cellulaire évolutionnaire quantique** où les automates cellulaires ont été évolués par des algorithmes génétiques quantiques.
- La mise en œuvre d'**une exécution parallèle** des automates cellulaires évolutionnaires quantiques qui permet d'atteindre une haute performance.

Organisation du mémoire

Notre mémoire est structuré de la manière suivante :

- Le chapitre I retrace un historique des automates cellulaires évolutionnaires, depuis leur invention par Norman Packard jusqu'aux développements plus récents. C'est le fruit d'une lecture approfondie d'un nombre assez-important des travaux qui existent dans le domaine afin d'extraire les informations les plus pertinentes.
- Le chapitre II détermine le cadre technique et formel du concept d'automate cellulaire évolutionnaire. Nous avons d'abord abordé le concept d'automate cellulaire puis celui des algorithmes génétiques et enfin le concept métis à partir de leur hybridation.

- Le chapitre III aborde la possibilité d'apporter quelques modifications sur le concept classique d'automate cellulaire évolutionnaire en l'étendant pour supporter l'évolution des automates cellulaires par des algorithmes génétiques quantiques. Des études théoriques ont été faites dans le but de déterminer jusqu'à quelle mesure ils peuvent s'avérer convenables à des traitements performants.
- Le chapitre IV est réservé à la description de l'implémentation de notre plateforme de simulation des automates cellulaires ainsi que la version parallèle des automates cellulaires évolutionnaires quantiques. Ce chapitre contient aussi un ensemble de tests expérimentaux qui permet l'évaluation de notre approche pour l'évolution des automates cellulaires.
- Enfin, nous terminerons par une conclusion générale où nous présenterons les leçons tirées à partir de nos expérimentations ainsi que les perspectives du travail réalisé dans le cadre de ce mémoire.

*« Le commencement de toutes les sciences,
c'est l'étonnement de ce que les choses sont ce qu'elles sont. »*

Aristote

Chapitre

1

Les automates cellulaires évolutionnaires : Etat de l'art

1. Introduction
2. La naissance des ACEV
3. La recherche autour du problème
 $\rho c = \frac{1}{2}$
4. La généralisation pour résoudre
d'autres problèmes
5. Conclusion

Les automates cellulaires évolutionnaires :

Etat de l'art

1. Introduction

Cette partie retrace l'évolution des automates cellulaires évolutionnaires (ACEV) depuis leur naissance jusqu'à nos jours. Nous avons choisi d'aborder le sujet en eu présentant un bref historique. Le but principal étant de construire une solide référence destinée à quiconque voulait étudier ce concept pour mieux l'appréhender et éventuellement pouvoir proposer des contributions concernant l'évolution des automates cellulaires (AC).

Nous avons distingué trois périodes essentielles :

- Invention des ACEV qui fut le fruit des travaux de Norman Packard, puis ceux de Melanie Mitchell sur le problème de la classification de densité.
- Recherches pour élucider le monde des ACEV.
- Exploration des espaces applicatifs des ACEV.

2. La naissance des automates cellulaires évolutifs

Les automates cellulaires évolutifs ont été inventés par Norman Packard à la fin des années 80 au Los Alamos National Laboratory (Etats-Unis). Norman Packard est un physicien américain, né en 1954 à Silver City, Nouveau-Mexique. Il était un ancien élève de *Reed College* et de l'*Université de Californie, Santa Cruz*. Il est considéré comme l'un des fondateurs de *la société de prédiction* et plus récemment la société *ProtoLife*.

La partie de ses travaux qui nous intéresse ici est sa contribution apportée sur l'évolution des AC. En 1988, Norman Packard publie un article intitulé " L'adaptation vers le bord du chaos " [13]. Il a enrichi la liste des concepts concernant les systèmes complexes par un autre nouvel appelé " *Le bord du chaos* ". Une des questions centrales concernant cette notion était de savoir si lorsque les systèmes biologiques doivent effectuer des calculs complexes pour survivre, le processus d'évolution en vertu de la sélection naturelle tend à sélectionner de tels systèmes près d'une transition de phase, autrement dit dans une zone frontière entre un comportement ordonné et un autre chaotique, en particulier, la capacité d'effectuer des calculs non triviaux, nécessite-t-elle un comportement dynamique qui soit prêt d'une transition vers le chaos ?

Packard a choisi d'aborder la question en modélisant des AC par des chromosomes puis les évoluer par un AG. Il a effectué des expérimentations complétant des travaux de recherches effectués par Wolfram puis par Langton sur les AC. Il semble que ses études sont incluses dans le cadre de définir la relation entre deux théories fondamentales pour l'étude des systèmes complexes sachant qu'elles sont historiquement indépendamment appliquées : la théorie de calcul et la théorie des systèmes dynamiques.

Depuis ces tous débuts l'étude des sciences dites de la complexité vise à comprendre les lois et les mécanismes par lesquels un comportement complexe global et cohérent peut émerger à partir d'une collection d'activités d'un nombre important de composants interagissant localement. Comme cette vaste catégorie de systèmes est principalement constituée d'une diversité systémique, les chercheurs voulant découvrir des points communs ou des lois universelles qui sous-tendent de tels systèmes doivent nécessairement étudier des cadres théoriques [*Framework*] qui soient très générales. Il paraît que la théorie de calcul et celle des systèmes dynamiques peuvent répondre à ces besoins sachant que celles-ci ont séparément

fourni de puissants outils pour la compréhension et l'exploration des propriétés d'un large éventail de tels systèmes [15].

D'une part, les scientifiques considèrent un système comme résolu s'il existe un ensemble fini d'expressions permettant de prédire son état à un instant t étant donné son état à un instant initial t_0 . Malheureusement, il n'est pas possible de déterminer une telle séquence pour un système complexe dû à son non linéarité (caractéristique fondamentale des systèmes complexes) ce qui rend son analyse et sa compréhension très difficiles. En faisant donc intervenir la théorie des systèmes dynamiques pour donner une description géométrique et topologique de la structure des ensembles des solutions. Autrement dit, elle donne une vue géométrique des éléments structurels d'un processus donné, des attracteurs et des séparateurs par exemple. Le but étant de déduire les structures génériques qui représentent les différents types des comportements typiques à travers le spectre des systèmes complexes [15].

D'autre part, l'adaptation de la théorie de calcul pour analyser un système complexe fournit un cadre pour décrire ses comportements. Si, par exemple, la correspondance globale entre l'état initial et l'état final est considérée comme un calcul, une des questions possibles est de déterminer quelle fonction a été calculée par la dynamique globale. Une autre question peut se poser : étant donné un système complexe, peut-il être conçu pour émuler une machine de Turing universelle ? D'autres questions peuvent également se poser [15].

Nous pouvons ainsi reformuler la question posée par Packard de façon plus générale : la capacité de calcul d'un système pour des traitements complexes de l'information, a-t-elle une relation avec d'autres mesures de comportements des systèmes ?

De là, nous pouvons sentir qu'il est nécessaire pour les chercheurs voulant étudier une telle relation de compter sur des modèles théoriques supportant les exigences précitées. Il semble que la classe du modèle abstrait d'AC est très utile pour répondre à ces besoins : une capacité pour exhiber différents types de comportements dynamiques et une capacité pour réaliser toute sorte de calcul. En outre, leur simulation par ordinateurs est relativement facile. Pour toutes ces raisons les AC ont toujours été considérés comme une bonne classe de modèles à utiliser pour étudier comment le comportement dynamique et la capacité de calcul sont liés dans les systèmes complexes.

2.1. La géométrie de l'espace des AC

L'histoire des ACEV tient ses origines depuis les travaux de recherches effectués par Stephen Wolfram sur les AC à l'*Institute for Advanced Study*. Stephen Wolfram est un physicien et mathématicien anglais, né à Londres, le 29 Août 1959. Il a publié un certain nombre d'articles sur la physique des particules en étant adolescent. A l'âge de vingt ans, il obtient son doctorat en physique des particules de l'université de *Cal Tech* et rejoint l'*Institute for Advanced Study* de *Princeton* en 1982. C'est là, en cherchant des modèles de la façon dont les galaxies s'étaient formées à partir d'un état initial chaotique qu'il s'intéressa aux AC [32].

C'est ce choix qui lui a valu d'être le premier chercheur qui a effectué une étude systématique de la totalité d'un espace d'AC. En 1984, Wolfram publie un article sous le titre de " L'universalité et la complexité des automates cellulaires " [8]. Il a proposé la première classification des AC selon leur **comportement dynamique**, en s'inspirant de la théorie des systèmes dynamiques. Partant d'une étude systématique des AC unidimensionnels à deux états sur un espace constitué de 256 AC, il a mis en exergue le résultat suivant :

Si chaque règle conduit à des motifs qui diffèrent dans le détail, tous les AC semblent pouvoir appartenir à seulement quatre classes qualitatives distinctes [8] [42].

- **Classe I** : L'évolution de l'automate après certains pas de temps, pour la quasi-totalité des différents états initiaux, tend vers un état homogène où les cellules ont la même valeur. Toute information existante sur l'état initial du système sera complètement détruite, la prédictibilité d'évolution est donc évidente. Partant de l'état initial, le système évolue toujours vers un état homogène.
- **Classe II** : L'évolution de l'automate conduit à un ensemble de structures stables ou périodiques (petite période), mais en tous cas simples et séparées. La prédictibilité d'évolution reste faisable. Les effets d'une cellule se propagent à un nombre fini de voisins. La modification d'une cellule de l'état initial n'affectera qu'une région finie de son entourage.
- **Classe III** : L'évolution de l'automate conduit à un motif chaotique caractérisé par des " *attracteurs étranges et des structures aperiodiques* ". Au cours de l'évolution, les cellules propagent les informations à vitesse constante, contrairement à ce qu'on peut observer dans les automates de la classe II. Connaître l'état d'une cellule après un nombre assez grand de pas de temps d'évolution exige la connaissance des états

initiaux d'un nombre très grand de cellules. La prédiction de l'évolution n'est donc possible qu'à partir d'un nombre infini d'états initiaux.

- **Classe IV** : L'évolution de l'automate conduit à un état mort en permettant l'apparition d'un petit nombre de structures complexes stables ou périodiques, parfois persistantes dans le temps. Le jeu de la vie en est le plus représentatif. Le degré de non-prédictibilité est encore plus important que dans les automates de la classe III. Cette classe est de loin la plus intéressante pour la Vie Artificielle.

Les trois premières classes ont été inspirées des catégories qui apparaissent dans l'étude des systèmes dynamiques, la quatrième étant la plus intéressante car elle est spécifique au domaine des AC. Cependant, il semble être très facile de détecter les insuffisances de cette classification du fait que la classe d'un automate ne peut être déterminée à priori, ici l'accent est mis sur ce que peut être un comportement dynamique d'un AC plutôt que toucher à comment l'aboutir. D'autres chercheurs ont également critiqué cette classification [32] en disant qu'il est impossible de décider si un automate appartient à la classe III ou à la classe IV sans intervention de la vue d'œil. D'ailleurs, même si l'on accepte l'hypothèse de Wolfram selon laquelle les automates de la classe IV sont supposés d'avoir la capacité de réaliser la calculabilité universelle, des résultats d'indécidabilité peuvent être obtenus : il serait alors impossible d'arriver à trouver une méthode systématique pour décider de la classe d'un AC [12]. Une conclusion est que prévoir la classe d'un AC à partir de sa fonction de transition reste encore un travail à faire.

2.2. Hypothèse de Langton (Le paramètre λ)

Deux ans plus tard, Chris Langton, ancien élève de Burks et inventeur d'un AC très fameux nommé *boucle de Langton* en hommage de son créateur, suit les traces de Wolfram et analyse l'exécution des milliers d'AC. Il a constaté que certains d'entre eux exhibent un comportement intéressant et complexe. Remarquant que la classification de Wolfram était phénoménologique, il a essayé de combler ce manque par l'introduction d'un paramètre appelé *Lambda* λ , permettant ainsi de cataloguer les règles qui produisent les types de dynamiques.

Rappelons que dans les systèmes dynamiques à état continu, il existe généralement un paramètre de contrôle variant continûment que l'on peut relier avec les changements qualitatifs du système en question et on parle alors de la bifurcation. Par analogie, Langton essaya de trouver un équivalent pour les AC. Il aboutit au choix du paramètre λ qu'il a défini

comme étant la fraction des règles pour lesquelles le nouvel état d'une cellule sera actif c'est-à-dire un état qui n'est pas en repos. De manière plus formelle, on exprime cela comme suit :

Soit un AC à k états, q est un état arbitrairement choisi d'être en '**repos**', λ est la fraction des règles pour lesquelles l'état de sortie est différent de q . L'automate du jeu de la vie a, par exemple, une valeur pour λ qui est égale à 0.273 .

Que se passe-t-il si : $\lambda = 0$, $\lambda = 1$, $\lambda = 1 - 1/k$?

- $\lambda = 0$: Toutes les transitions mènent à l'état de repos.
- $\lambda = 1$: Pas de transition vers l'état de repos.
- $\lambda = 1 - 1/k$: Les états sont également représentés dans les règles de transitions (une fraction de $1/k$ correspond à l'état de repos).

Un premier obstacle est surpassé, mais il restait à caractériser **le comportement dynamique** en fonction de λ . Pour ce faire, Langton a formé un ensemble d'échantillons d'AC à deux dimensions composés selon la méthode de Monte Carlo puis il a analysé les « *comportements moyens* », c'est-à-dire les plus probablement observés. Chaque configuration initiale a été évoluée par un sous-espace d'AC ayant une même valeur donnée à λ . Langton a utilisé diverses statistiques comme l'information mutuelle et la longueur des transitoires afin de classer le comportement moyen à chaque valeur de λ . En vertu de son observation, il a rapidement constaté que, comme λ est graduellement incrémenté de 0 à $[1 - 1/k]$, le comportement moyen des règles passe par les régimes suivants :

Point fixe → *périodique* → « *complexes* » → *chaotique*.

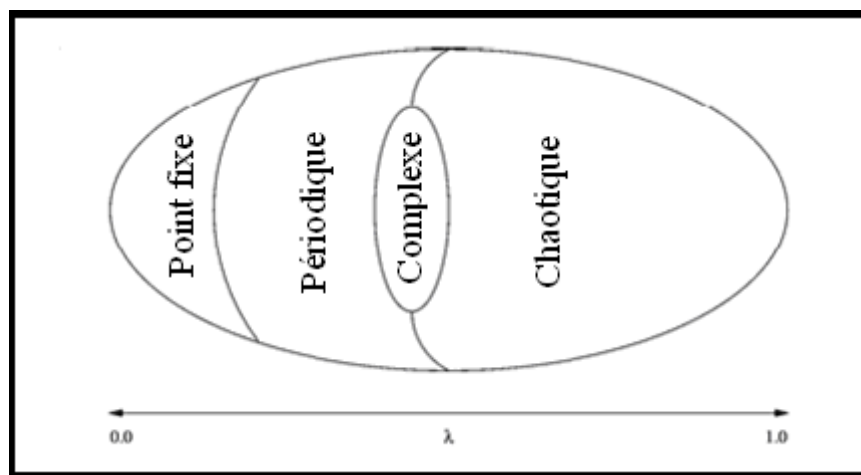


Figure 1.1. Structure d'un espace d'AC en fonction de λ

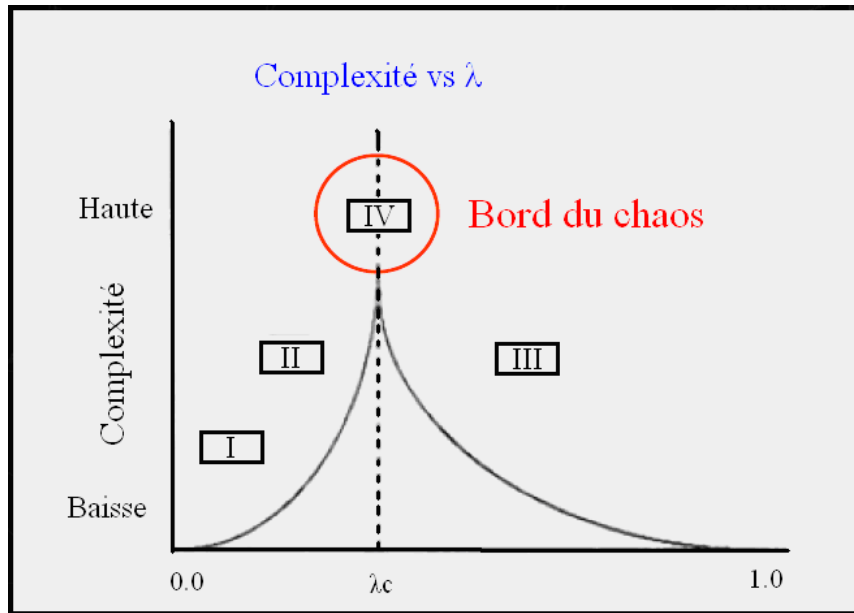


Figure 1.2. Variations de la complexité en fonction de λ

Ce sont donc les mêmes quatre classes d'AC établies par Wolfram mais figurant par l'ordre : 1, 2, 4, 3. L'analyse des différentes mesures de la complexité a montré qu'elle apparaît à mi-chemin entre l'ordre et le désordre. Langton n'a pas observé un désordre qui s'accroît continûment à partir de l'ordre mais une complexité qui augmente jusqu'à un seuil - *une transition de phase* - où elle décroît ensuite, comme si l'on est en présence *d'un système physique*. En effet, les physiciens ont montré que les systèmes physiques subissent des modifications profondes de certaines de leurs propriétés pendant une transition de phase où la complexité sera maximale et on se retrouve alors loin de l'équilibre, en opposition de ce qui se passe avant et après où il y a présence d'une certaine forme de stabilité dynamique. La question qui se pose ici est : jusqu'à quel niveau de complexité peut-on aller ? Cela revient à étudier la capacité de traitement de l'information du système [51].

Langton a supposé que les systèmes physiques dont l'état se trouvant près des transitions de phases continues, c'est-à-dire près de la *criticité*, entre un état ordonné (solide) et un autre chaotique (liquide) sont surtout capables de faire des calculs. Par analogie, il expliqua le phénomène observé en disant que seuls les AC de la classe IV peuvent faire des calculs complexes comme dans le jeu de la vie qui possède la propriété de la calculabilité universelle et qui est donc potentiellement capable de traiter n'importe quel problème. Selon Langton, seuls les AC de la classe IV peuvent avoir à la fois des caractéristiques des classes I et II (stockage de l'information sous forme d'attracteurs) et de la classe III (transmission de l'information en détruisant l'état antérieur). Il est même allé jusqu'à théoriser que les systèmes

vivants et d'autres systèmes auto-organisés montrent des caractéristiques qualitatives associées à l'intermède entre la périodicité et le chaos. Ceci a conduit à émettre une thèse dont la portée philosophique :

« La vie, symbolisée par la classe IV, serait un phénomène susceptible de se trouver en l'ordre, symbolisé par la classe II et le chaos, symbolisé par la classe III. Pour généraliser encore, il semblerait que l'émergence de la vie dans l'univers ne soit possible que grâce à un ajustement exceptionnel des lois fondamentales de cet univers, le plaçant à la frontière entre l'ordre et le chaos. » [32]

Dans tous les cas ceci reste une question de point de vue dont le débat n'est pas une de nos préoccupations premières. Langton a donc émis l'idée que *les AC capables d'effectuer des calculs complexes sont trouvés dans une transition de phase d'un comportement ordonné vers un autre chaotique.*

Effectivement, on peut sentir le besoin d'un paramètre décrivant le comportement d'un AC à partir de sa fonction de transition. Packard, ancien collègue de Langton, a fait des expérimentations visant à déterminer des valeurs brutes de λ_c par l'étude des AC unidimensionnels à deux états (ayant $r = 3$, r est le rayon du voisinage). La corrélation de son paramètre γ , mesure d'imprévisibilité dans les figures espace-temps, en fonction de λ a permis la découverte de deux valeurs pour λ_c : $\lambda_c \approx 0.83$ et $\lambda_c \approx 0.23$. Pour Packard, ces résultats ont été sans doute comme la confirmation de l'hypothèse de Langton.

Cependant, il paraît très clair que ni Langton ni Packard n'ont corrélié λ en fonction d'une mesure indépendante de calcul ce qui peut laisser planer un doute. Packard a trouvé la solution à cette impasse en **évoluant des populations d'AC par un AG pour effectuer un calcul particulier : celui de la classification de densité [13].**

2.3. L'idée de départ de Packard

Packard a fait ses expérimentations sur le problème dit de « *La classification de densité* » ou encore « *ρ -classification task* ». Ce problème peut se résumer en :

Soit une configuration initiale constituée de cellules dont chacune d'entre elles peut se trouver dans deux états : 0 ou 1. La tâche de calcul que doit effectuer l'AC est de décider si la configuration initiale contient plus de la moitié de cellules se trouvant dans l'état 1. On dénote par ρ la densité des cellules qui se trouvent dans l'état 1, et par ρ_c le seuil de densité pour le

classement, dans ce cas on parle de $\rho_c = 1/2$. Nous allons exposer ce problème en détail dans une section suivante.

Il ne restait donc qu'à choisir un mécanisme pour l'évolution, il a ainsi utilisé un AG évoluant une population d'AC, le but était double, tester la validité de deux hypothèses :

- Les AC capables d'effectuer des calculs complexes sont les plus susceptibles d'être trouvés près des valeurs de λ_c .
- Lorsque les AC sont évolués pour effectuer un calcul complexe, l'évolution aura tendance à choisir des règles près des valeurs de λ_c .

Son AG a démarré avec une population initiale de fonctions de transitions générées aléatoirement, cependant elles ont été contraintes d'être uniformément réparties à travers les valeurs de λ entre 0 et 1. Chaque fonction dans la population a été évaluée afin de déduire sa capacité pour effectuer la tâche demandée en lançant son exécution sur une configuration initiale générée aléatoirement pour un certain nombre de pas de temps puis en mesurant la fraction des cellules dans le réseau qui ont eu un état correct : pour les configurations initiales où $\rho > \rho_c$ l'état final correct pour chaque cellule est de 1 et exactement l'inverse pour celles ayant $\rho < \rho_c$ où l'état final correct pour chaque cellule est de 0. Si, par exemple, un AC a été exécuté sur une configuration initiale avec $\rho > \rho_c$ et à la fin d'exécution le réseau contenait 70 % de cellules se trouvant dans l'état 1, alors le score de cet AC serait 0.7. Selon le principe des AG, l'évolution se fait génération après génération, pour chacune les individus de la population sont classés selon leurs *valeurs d'adaptation (Fitness)* puis des opérations génétiques sont effectuées dans le but de former la génération suivante.

Packard a exécuté son algorithme sur environ cent générations initiales puis il a fusionné tous les individus des populations résultantes à partir des différentes exécutions pour en former une seule les englobant. A ce moment là, la corrélation entre λ et la tâche réalisée peut maintenant s'effectuer. Il a ainsi considéré la fréquence des règles observées dans la population de tous les AC pour la corréler en fonction de λ . Il en résulte que les règles se regroupèrent autour de deux régions de λ_c . En comparant ces résultats avec ceux précédemment trouvés (corrélation de la mesure d'imprévisibilité γ), Packard a montré que les deux résultats coïncident permettant par conséquent de confirmer les hypothèses testées, Mitchell écrit ainsi :

« Packard interpreted these results as evidence for the hypothesis that, when an ability for complex computation is required, evolution tends to select rules near the transition to chaos. He argues, like Langton, that this result intuitively makes sense because “rules near the transition to chaos have the capability to selectively communicate information with complex structures in space-time, thus enabling computation. » [16]

[Packard a interprété ces résultats comme la preuve de l'hypothèse selon laquelle, quand une capacité de calcul complexe est requise, **l'évolution** tend à sélectionner des règles près d'une transition vers le chaos. Il fait valoir, comme Langton, que ce résultat est intuitivement logique car **les règles près de la transition vers le chaos ont la capacité de communiquer l'information de manière sélective avec des structures complexes dans leurs figures espace-temps, permettant ainsi le calcul.**] [Ma TRD].

La conclusion de Packard a donc enlevé l'incertitude de l'hypothèse de Langton. C'est là, en créant une nouvelle approche pour l'évolution des AC par des AG ce que l'on connaît maintenant sous le nom **d'automate cellulaire évolutionnaire**. Bien que son idée soit à haut degré d'utilité et d'efficacité, elle n'a réellement été exploitée qu'après l'an 1993 en coïncidence avec la première grande publication, par l'équipe de recherche « *The Evolving Cellular Automata Group* » dirigée par Mitchell, chargée de rendre les AG comme un outil très populaire pour évoluer des AC [16]. Le but étant la résolution du problème de la classification de densité. En quoi consisterait ce problème ?

2.4. Le problème de la classification de densité

Le problème dit de *la classification de densité*, ou encore appelé *problème de majorité*, est l'un des problèmes qui restaient insolubles pendant des années. Il sert à trouver un AC unidimensionnel qui exécute fidèlement le vote à la majorité. Etant donné une configuration d'un AC à deux états avec $i + j = N$ cellules au total, i étant le nombre de cellules qui se trouvent dans l'état 0 tandis que j est le nombre de cellules qui se trouvent dans l'état 1, ρ est un seuil donné. Une solution correcte au problème de majorité doit finalement fixer toutes les cellules à 0 si $i / N < \rho$ et à 1 si $i / N > \rho$. On peut légitimement se demander pourquoi cette tâche est considérée comme étant un calcul non trivial pour les AC ayant un petit rayon ($r \ll N$).

Contrairement aux systèmes ayant un contrôle central ou un support de stockage d'information central (d'ailleurs les ordinateurs sont un excellent exemple de tels systèmes), il semble clairement que la densité est une propriété globale, et il est évident que les cellules constituant une configuration ne peuvent connaître le nombre total des 0 qui se trouvent dans le système du fait que seules les interactions locales entre elles sont permises, comment donc chaque cellule pourrait être capable de déterminer s'il y a une majorité de 1 par rapport aux 0 partant seulement d'un ensemble d'interactions locales ? Il faut nécessairement que les cellules doivent communiquer l'information entre elles jusqu'à ce que la tâche soit réalisée.

Il semble que Packard et Langton n'ont pas rigoureusement défini cette tâche pour qu'un traitement complexe de l'information soit supporté du fait qu'ils n'ont pas mentionné comment le calcul complexe sera réalisé et cela pourrait être un peu problématique. Mitchell a donc comblé ce manque en redéfinissant de nouveau cette tâche afin que les exigences d'un traitement complexe soient supportées :

« Though this term was not rigorously defined in [17] or [24]¹. One possible definition might be any computation for which the memory requirement increases with N (i.e., any computation which corresponds to the recognition of a non-regular language) and in which information must be transmitted over significant space-time distances (on the order of N). Under this definition the $pc = 1/2$ classification task can be thought of as a nontrivial computation for a small radius CA. The effective minimum amount of memory is proportional to $\log(N)$ since the equivalent of a counter register is required to track the excess of 1's in a serial scan of the initial pattern. And since the 1's can be distributed throughout the lattice, information transfer over long space-time distances must occur. This is supported in a CA by the non-local interactions among many different neighborhoods after some period of time. » [16]

[Bien que ce terme n'est pas rigoureusement défini dans [17] ou [24]. Une définition possible serait peut-être tout calcul pour lequel l'exigence en mémoire

¹ [17] correspond à l'article de Langton intitulé « Computation at the edge of chaos: Phase transitions and emergent computation » (1992), [24] correspond à la référence N° 13 de notre bibliographie.

augmente avec N (c'est-à-dire, tout calcul qui correspond à la reconnaissance d'un langage non-régulier) et dont les informations doivent être transmises sur de longues distances dans les diagrammes espace-temps (de l'ordre de N). Aux termes de cette définition, la tâche de classification $pc = 1/2$ peut être pensée comme un calcul non trivial pour un AC ayant un petit rayon. Le montant minimum effectif de mémoire est proportionnel à $\log(N)$ d'où l'équivalent d'un registre compteur est nécessaire pour suivre l'excès des 1 en utilisant un balayage en série dans la configuration initiale. Et comme les 1 peuvent être répartis à travers le réseau, le transfert d'informations sur de longues distances dans les digrammes espace-temps doit se produire. Ceci est supporté par un AC en utilisant des interactions non-locales entre les différents voisinages après une certaine période de temps] [Ma TRD].

En 1978, Gacs, Kurdyumov et Levin proposent la première solution à ce problème [7], l'état de chaque cellule est calculé comme suit :

Pour chaque cellule qui se trouve dans l'état 0, son prochain état est déterminé par la majorité entre les valeurs d'elle-même, de son voisin immédiat à gauche, et de son voisin situé dans la troisième position à gauche, et c'est par symétrie pour les cellules qui se trouvent dans l'état 1, le prochain état est déterminé par la majorité entre les valeurs d'elle-même, son voisin immédiat à droite, et son voisin situé dans la troisième position à droite.

Cette solution n'a pu résoudre ce problème que partiellement, c'est-à-dire sur un sous ensemble de configurations mais pas sur la totalité de l'espace des configurations possibles. Initialement, cette règle a été inventée dans le cadre de leurs études concernant le calcul fiable dans les AC et elle n'était donc pas concernée par un objectif de classement. Sa seule particularité est que **la plupart des configurations** aboutissent à un point fixe correspondant à la densité des 1 avec une faible erreur. Packard s'est donc inspiré de cette règle pour évoluer des AC afin de réaliser une tâche de classement.

2.5. L'institut de Santa Fe (SFI) réexamine les résultats de Packard

L'institut de Santa Fe (Santa Fe Institute ou SFI) est un institut de recherche dédié à l'étude des systèmes complexes. Il se situe à Santa Fe (Nouveau-Mexique), fondé par un ensemble de chercheurs en 1984. Bien qu'il ait récemment vu le jour, les travaux réalisés au sein de cet institut sont d'une grande importance. La partie des travaux qui nous intéresse ici est celle

traitant l'évolution des AC. En 1993, l'équipe de recherche « The Evolving Cellular Automata Group » annonce un avis différent de celui de Packard : après avoir réexaminé ses hypothèses, elle n'a tardé de déclarer ses interprétations des résultats obtenus incorrectes.

Une série d'expérimentations semblables à celles qui viennent d'être décrites a été effectuée mais les résultats étaient remarquablement différents, et pourtant elle a été tenue à répéter les mêmes conditions expérimentales. Ce groupe n'a pas trouvé des règles regroupées autour des deux régions de λc trouvées par Packard, au contraire, elles se regroupèrent plus proches de la région où $\lambda \approx 0.5$. Outre ces expérimentations, d'autres études théoriques ont suggéré que les règles les plus réussies pour la réalisation d'une tâche de classement donnée seront proches d'une valeur particulière de λ qui dépend de la valeur de ρc . Ainsi, pour cette classe de tâches de calcul, les valeurs de λc qui sont associées à un "bord du chaos" ne sont pas en corrélation avec la capacité des règles pour accomplir de telles tâches. Une conclusion est que le test d'origine n'a pas donné des preuves des hypothèses pour lesquelles était destiné à tester. Mais en tout cas, on ne peut affirmer l'inexactitude des hypothèses de Langton et de Packard. Mitchell écrit ainsi :

« The results presented here do not disprove the hypothesis that computational capability can be correlated with phase transitions in CA rule space. Indeed, this general phenomena has already been noted for other dynamical systems. More generally, the computational capacity of evolving systems may very well require dynamical properties characteristic of phase transitions if they are to increase their complexity. We have shown only that the published experimental support cited for hypotheses relating λc and computational capability in CA was not reproduced. »
[16]

[Les résultats présentés ici ne réfutent pas l'hypothèse selon laquelle la capacité de calcul peut être corrélée avec des transitions de phase dans l'espace des AC. En effet, ce phénomène général a déjà été noté pour d'autres systèmes dynamiques. Plus généralement, la capacité de calcul des systèmes évolutifs peut bien exiger des caractéristiques de propriétés dynamiques pour des transitions de phases s'ils augmentent leur complexité. Nous avons seulement montré que la publication du support expérimental cité [c'est-à-dire celui de Packard] pour les hypothèses relatives à λc et la capacité de calcul dans le cadre des AC n'a pas été reproduite]
[Ma TRD].

En effet, Langton lui-même a admis que l'hypothèse qu'il a émise n'était pas toujours en adéquation avec les résultats observés. D'ailleurs, selon Mitchell et après analyse des études statistiques de Langton et de Packard, la plupart des valeurs de λ ont montré une variance très élevée. Ceci implique que le comportement de toute règle particulière ayant une valeur donnée à λ pourrait être très différent du comportement moyen à cette valeur. L'utilisation de ce paramètre comme outil de prévision de la dynamique est donc moins claire notamment pour les AC à petit rayon de voisinage.

Cette conclusion n'était qu'un premier pas vers la découverte de nouveaux horizons, des structures si complexes apparaissant dans les figures espace-temps résultants. Quelques unes des règles découvertes par évolution des AC ont montré l'existence des stratégies sophistiquées de calcul émergent. Il paraissait alors que l'évolution des AC constitue une bonne voie pour faire des recherches pouvant mener plus loin. L'étude du problème de la classification de densité va donc prendre une nouvelle dimension en jouant le rôle d'un problème type largement discuté. Après avoir obtenu tous ces résultats, l'institut de Santa Fe annonce la naissance d'un grand projet de recherche dans le but d'explorer l'univers des AC en utilisant les AG. L'objectif principal étant la découverte de nouvelles règles montrant l'existence d'un calcul émergent nécessitant une coordination globale.

L'une des raisons ayant motivé sa naissance était de comprendre comment l'évolution naturelle crée des systèmes dans lesquels un calcul émergent s'effectue, c'est-à-dire, partant seulement d'actions locales des composants simples, il sera possible d'aboutir à une coordination globale de traitement de l'information ; les colonies d'insectes, les systèmes économiques, le système immunitaire et le cerveau ont tous été cités comme exemples de tels systèmes où l'émergence des calculs survient. Toutefois, il n'est pas bien compris la manière dont ces systèmes naturels effectuent des calculs [21].

Une autre motivation paraissant très intéressante pour le domaine d'ingénierie. Les AC constituent un nouveau paradigme caractérisé principalement par un traitement des problèmes selon une approche ascendante (du simple vers le complexe), parallèle et en déterminant les comportements des entités élémentaires de façon locale. Cependant, la difficulté de maîtriser le comportement émergent des AC ou leur conception pour exhiber un comportement visé ou accomplir une tâche particulière a jusqu'à présent limité leurs applications en science, en ingénierie et plus généralement pour faire des calculs. L'automatisation du processus de la conception (programmation), via les AG, pourrait donc renforcer grandement l'amélioration

de la viabilité des AC. Ceci peut conduire à trouver des moyens pour l'ingénierie des calculs émergents sophistiqués dans les systèmes multiprocesseurs décentralisés [21].

3. La recherche autour du problème $\rho c = 1/2$

Malgré toutes les maintes reprises de Packard puis de Mitchell et ses collègues pour résoudre le problème de la classification de densité $\rho c = 1/2$, cela fut malheureusement en vain du fait qu'aucun des résultats obtenus n'a montré sa réalisation parfaite pour toutes les configurations possibles. Ce problème pouvait-il être parfaitement résolu ou cela n'était qu'une pierre philosophale ? Existe-t-il vraiment un AC pouvant effectuer cette tâche parfaitement ?

Effectivement, il découle d'une simple observation que le modèle classique d'AC est essentiellement programmé à un très bas niveau où une seule règle est cherchée afin d'être appliquée à toutes les cellules [14], une tâche qui peut être pénible même avec une approche évolutive. Ainsi, les résultats de Packard puis de Mitchell et ses collègues, nous font sentir que, comme toute classe de modèles, leur utilisation possède des limites. Dans leur conception classique, ils sont adaptés à la modélisation des flux d'information [32]. Pour leur faire jouer le rôle d'outil de résolution des problèmes calculatoires très complexes, il faut nécessairement procéder à des modifications du modèle de base, en utilisant plus d'une règle locale par exemple.

Cette conclusion a été confirmée en 1995 par Land et Belew qui aboutissent à décider de l'impossibilité de réaliser cette tâche [20], c'était la preuve qu'aucun AC unidimensionnel à deux états n'est capable de la réaliser parfaitement pour toutes les configurations possibles. Cette démonstration a marqué un tournant dans l'étude de ce problème car contrairement à ce qu'il a été pensé, elle a montré l'impossibilité de trouver des règles locales pouvant le résoudre parfaitement. Les chercheurs après cette publication ont par conséquent été contraints d'abandonner l'idée d'évoluer des AC par des simples AE pour trouver une seule simple règle conduisant à la résolution parfaite de ce problème.

Le début fut avec le fameux Moshe Sipper, ancien chercheur à l'école polytechnique fédérale de Lausanne en Suisse. Remarquant la rigidité des AC uniformes, Sipper a émis l'idée que l'augmentation des performances peut s'effectuer en faisant plus simple. Ce qui compte en définitive, c'est l'efficacité dans l'achèvement de la tâche demandée. Il a ainsi proposé d'utiliser un autre type d'AC : les AC dits de *non-uniformes* [22], où à chaque cellule peut correspondre une règle différente. Ainsi, il paraît que la non-uniformité sert à assouplir les

restrictions sur les règles appliquées aux cellules. Par ailleurs, il est clair que l'espace de recherche s'agrandit énormément ce qui le rend très difficile à explorer. On peut donc dire que trouver l'AC non-uniforme capable de réaliser la tâche de la classification de densité sera de plus en plus difficile même si l'on accepte que Sipper a considéré que la grandeur de l'espace de recherche ne doit pas être prise en tant qu'une entrave comme il semble à première vue, mais permet d'engendrer de nouvelles voies d'évolution aboutissant à des systèmes hautement performants.

Son algorithme a omis la notion d'évolution globale d'une population de solutions, la remplaçant par *la co-évolution locale* d'un ensemble de règles positionnées sur une grille constituant un même AC non-uniforme. Son algorithme était différent de celui de Packard et de celui de Mitchell et ses collègues en deux points essentiels :

- La fonction *Fitness* dans l'algorithme de Sipper a été calculée au niveau cellulaire où l'on accumule pour chaque cellule le score marqué sur les configurations initiales évoluées, ce qui s'oppose avec la *Fitness* utilisée dans l'algorithme de Packard où elle a été calculée de façon globale.
- Le processus évolutif est effectué de manière complètement locale où les opérateurs génétiques ne sont appliqués qu'entre les cellules directement connectées, contrairement à ceux de Packard qui nécessitent un ordonnancement des individus pour effectuer des opérations génétiques.

Ses expérimentations ont montré que pour les AC non-uniformes ayant $r = 3$ la meilleure fitness obtenue était de 0.92 alors que celle des AC ayant $r = 1$ était de 0.93 (r est le rayon du voisinage). En comparant ces résultats avec ceux trouvés par des AC uniformes (0.95 et 0.83 respectivement), Sipper a considéré que ceci est notable en tenant compte de la grandeur des espaces de recherches concernés qui sont beaucoup plus larges que ceux des AC uniformes. Il a ainsi mis en exergue les résultats suivants :

- Les AC non uniformes peuvent atteindre une haute performance pour effectuer des calculs non triviaux.
- De tels systèmes peuvent être évolués plutôt que conçus.

Presque un an plus tard, Sipper propose le premier manuel consacré au problème de l'évolution des AC et publie *Evolution of Parallel Cellular Machines : The Cellular Programming Approach* [24]. Les résultats ont été présentés de façon nettement concise, permettant ainsi aux chercheurs et aux étudiants de mieux se familiariser avec le domaine.

Ce n'est qu'en 1997 que la première solution à ce problème est enfin réalisée, par Henryk Fuks', qui complète les travaux inachevés et publie l'article « Solution of the Density Classification Problem with Two Cellular Automata Rules » [25]. Le point de départ étant la conclusion de Belew et Land selon laquelle une simple règle ne peut à elle seule résoudre le problème. Fuks' a trouvé la solution en comptant sur plus d'une règle, il a procédé comme un naturaliste en s'inspirant du phénomène selon lequel de nombreuses entités naturelles doivent créer des sous groupes spécialisés à une sous tâche pour accomplir une tâche globale. Il a montré que l'application successive d'une paire de règles écrites à la main effectue cette tâche parfaitement. La première règle est dite de « *la règle de circulation 184* » [de l'anglais : *Traffic Rule*], alors que la deuxième est dite de « *la règle de majorité 232* » [de l'anglais : *Majority Rule*].

Fuks' a donc souligné que cela pourrait être accompli comme une '**chaîne de montage**'. Il écrit ainsi :

« If we think about the cellular automaton as a model of a multicellular organism composed of identical cells, or a single kind of 'tissue,' we could say that evolution reached a "dead end" here. In the biological evolution, when the single-tissue organism cannot be improved any further, the next step is the differentiation of cells, or aggregation of cells into organs adapted to perform a specific function. In a colony of insects, this can be interpreted as a 'division of labor,' when separate groups of insects perform different partial tasks. For cellular automata, this could be realized as an 'assembly line,' with two (or more) different CA rules: the first rule is iterated t_1 times, and then the resulting configuration is processed by another rule iterated t_2 times. Each rule plays the role of a separate 'organ,' thus we can expect that such a system will be able to perform complex computational tasks much better than just a single rule. » [25]

[Si nous pensons aux AC comme un modèle d'un organisme multicellulaire composé de cellules identiques, ou une seule sorte de 'tissu', on pourrait dire que l'évolution ici a atteint une 'impasse'. Dans l'évolution biologique, lorsque l'organisme à tissus

unique ne peut pas être amélioré plus loin, l'étape suivante est la différenciation des cellules, ou agrégation des cellules en des organes adaptés pour effectuer une fonction spécifique. Dans une colonie d'insectes, ceci peut être interprété comme une 'division du travail', lorsque des groupes séparés d'insectes effectuent différentes tâches partielles. Pour les AC, ceci pourrait être réalisé comme une 'chaîne de montage', avec deux règles de transitions différentes (ou plus) : la première règle est itérée t_1 fois, puis la configuration résultante est traitée par une autre règle itérée t_2 fois. Chaque règle joue le rôle d'un 'organe' séparé, on peut donc s'attendre à ce qu'un tel système soit en mesure d'effectuer des tâches de calcul complexe, beaucoup mieux que juste une règle unique.] [Ma TRD].

Petit à petit, l'étude de ce problème dépasse les limites tracées par Mitchell et Sipper afin de jouer le rôle d'un problème type utilisé dans l'étude des *systèmes co-évolutifs*. Dans la nature, les interactions entre les prédateurs et les proies fournissent une force motrice vers la complexité. Cela parce qu'il y a une forte pression d'évolution pour les proies afin de mieux se défendre elles-mêmes. Les futures générations des prédateurs doivent donc développer des stratégies pour mieux les attaquer. Cette course aux armements se caractérise par une interaction d'aptitude inverse : le succès d'un côté est ressenti par l'autre côté comme un échec afin de maintenir les chances de survie.

Les biologistes utilisent le terme "*hypothèse de la Reine Rouge*" pour désigner ce phénomène où un changement constant est nécessaire pour survivre. Ce nom vient du livre "*Alice au pays des merveilles*" au cours duquel le personnage principal et la Reine Rouge se lancent dans une course effrénée. Alice demande alors : « *Mais, Reine Rouge, c'est étrange, nous courons vite et le paysage autour de nous ne change pas ?* » Et la reine de répondre : « *Nous courons pour rester à la même place.* » [43]. C'était Van Valen qui a présenté à l'an 1973 le terme dans le contexte de la course aux armements biologiques. Comme d'habitude, les chercheurs en informatique ne cessent de s'inspirer des phénomènes biologiques. Cette notion a été utilisée dans la construction d'une *approche basée sur la co-évolution pour résoudre des problèmes d'optimisation* où deux populations peuvent être évoluées en tant que *prédateurs et proies*.

En observant que l'erreur d'échantillonnage qui résulte de la sélection aléatoire des configurations initiales réduit souvent l'efficacité du processus évolutif, Paredis, qui a longtemps étudié la résolution des problèmes par co-évolution, introduit une modification

profonde sur le processus évolutionnaire établi par Mitchell pour la résolution du problème de la classification de densité [26]. Il a proposé d'utiliser *la co-évolution* comme stratégie d'évolution à la place de celle adoptée par un AG. Il a ainsi créé une population de prédateurs, représentée par des AC, attaquant une autre population de proies, représentée par des configurations initiales. Un tel algorithme utilise généralement deux fonctions d'évaluations différentes : l'une pour les prédateurs et **son inverse** pour les proies.

Mais assez étonnamment, les résultats ont été contraires de ce qu'il était attendu, des performances extrêmement pauvres ont été obtenues. En effet, le fait que l'aptitude d'un individu d'une population dépend des individus de l'autre population donne lieu à un environnement constamment changé dans lequel les deux populations tentent de maximiser leurs chances de survie. En fait, l'interaction de la fitness entre les deux populations signifie que le succès d'un côté est un échec pour l'autre côté. C'est la raison pour laquelle les configurations rendent la vie difficile pour les AC et, par conséquent, obtenir une bonne rentabilité. Une leçon tirée de cette étude est que la prudence est nécessaire lors de l'utilisation de la co-évolution pour résoudre des problèmes d'optimisation et surtout quand il n'y a pas de solution satisfaisant tous les tests possibles qui pourrait être le cas ici. Paredis était en fin de compte contraint d'éliminer l'évolution dans la population des proies pour la garantir dans celle des prédateurs ce qui lui a permis d'aboutir à une performance plus élevée.

En 1998, et après un examen profond des précautions émises par Paredis lors de l'utilisation de la co-évolution, Pollack définit un nouveau concept dit de *l'apprentissage par co-évolution* [28], fondé sur la combinaison de deux techniques : *la co-évolution et le partage de ressources*. Il a introduit une procédure de recherche qui traite avec succès les obstacles entravant l'efficacité du mécanisme de la co-évolution. L'objectif principal était d'éviter l'effet de la Reine Rouge dans les systèmes co-évolutifs. La démonstration de la puissance de son algorithme a été faite en considérant le problème de la classification de densité comme un problème cible. Ainsi, deux populations ont été évoluées : une pour les règles de transitions et l'autre pour les configurations initiales. Les résultats expérimentaux ont conduit à la découverte de deux nouvelles règles montrant une très nette amélioration par rapport aux règles précédemment connues. Pollack a interprété ce résultat par la puissance du mécanisme de la co-évolution qui a permis d'atteindre une haute performance lorsqu'il a été correctement utilisé. Cet algorithme, est-il testé pour résoudre d'autres problèmes ? La source de cette performance est-elle due à la co-évolution ? La réponse à ces questions est loin d'être

affirmative. L'algorithme de Pollack toutefois, a été conçu pour être général dans la mesure où il capture les caractéristiques importantes de l'évolution des systèmes dont la coordination globale émerge seulement quand les interactions locales sont possibles mais pourtant il n'a été testé que seulement sur le problème de la classification de densité. Il serait donc prématuré d'assurer que la co-évolution est la source d'une telle performance et on ne peut même pas vérifier cette généralité.

Un an à peu près plus tard, Justin Werfel, collègue de Mitchell au laboratoire du SFI, réexamine les travaux de Pollack effectués sur la co-évolution des AC. Comme Paredis a montré qu'une simple version d'un algorithme co-évolutif n'a parvenu, à elle seule, à produire des stratégies performantes pour la tâche de la classification de densité, il semblait naturel de se demander si le succès de l'algorithme de Pollack est davantage dû à la co-évolution ou à un partage de ressources, ou à une combinaison particulière des deux [29].

La démonstration présentée par Justin Werfel a suggéré que, quand les performances d'un AG sont améliorées par une combinaison de partage de ressources avec la co-évolution, l'amélioration est, en grande partie (mais pas entièrement), en raison du partage de ressources, plutôt que de la co-évolution de sa propre initiative. Cela contredit la conclusion offerte par Pollack et selon laquelle la co-évolution a été la force motrice de l'amélioration de la performance de son AG. Il a également présenté la preuve que le partage de ressources fonctionne de façon préservant la diversité de la population lors de l'évolution des AC [29].

Après ce rapide tour d'horizon théorique et après être prouvé qu'aucun AC uniforme à lui seul n'est capable d'effectuer cette tâche parfaitement, il ne reste alors qu'une seule voie probablement qui conduit vers sa résolution définitive : chercher un AC non-uniforme pouvant l'effectuer. Pouvait-il exister un AC non-uniforme capable de résoudre ce problème parfaitement ?

Signalons qu'en 2002, Mathieu Capcarrère, ancien élève de Sipper à l'EPFL, a prouvé en étant un doctorant que, pour cette version du problème, aucun AC non-uniforme à deux états quelque soit son rayon de voisinage, ne peut le résoudre parfaitement. Ainsi, cette tâche paraissait totalement impossible pour les AC à deux états (uniforme et non uniforme). En effet, à ce jour nous n'avons connaissance d'aucun AC (désigné ou évolué) quelque soit sa nature (uniforme ou non-uniforme) qui effectue cette tâche parfaitement et les chercheurs ne

font qu'améliorer seulement les performances globales sans aboutir à une solution définitive. Cette conclusion a donc mis une fin à cette problématique.

Par ailleurs, Mathieu Capcarrère a prouvé en 1996 [23] que si l'on redéfinit la configuration finale désirée dans le problème de la classification de densité, c'est-à-dire l'évolution de l'automate va tendre vers une configuration finale autre qu'un point fixe ; il existe un AC élémentaire noté par *184* selon la notation de Wolfram qui effectue cette tâche parfaitement. Cependant, sous cette nouvelle définition cette tâche perd spectaculairement son rôle en termes de calculabilité.

Etant donné tous ces efforts dépensés pour résoudre un tel problème, peut-on juger que la conclusion de Land et Belew et celle de Mathieu étaient une déception ? Ce jugement sévère était-il justifié ? Selon Kuhn, une discipline scientifique doit apporter des énigmes à la communauté scientifique concernée [32]. Nous pouvons donc estimer jusqu'à quel niveau de '*scientificité*' peut-on aller en étudiant le problème de la classification de densité.

Le premier problème fut posé au moment de la considération de la tâche de la classification de densité comme une tâche *non triviale* par Packard en 1988. La révision de ses travaux par Mitchell a suggéré que ce terme n'était pas rigoureusement défini car sa définition était basée sur la règle GKL qui n'était en réalité inventée que pour trouver un AC, dont le comportement est robuste aux petites erreurs dans la règle de mise à jour des configurations. Mitchell l'a donc redéfini autrement pour qu'un traitement complexe de l'information soit supporté par l'étude de ce problème en examinant les figures espace-temps. Après cette redéfinition, quelques-unes des stratégies de calcul émergent ont été découvertes par évolution d'AC. Citons comme exemples, les stratégies découvertes par Mitchell, par Das, et enfin par Pollack. Toutes ces stratégies calculatoires ont contenu un traitement complexe de l'information dans leurs diagrammes espace-temps.

Un autre problème qui a occupé la communauté des chercheurs fut celui de l'analyse des diagrammes espace-temps. Afin d'analyser le comportement résultant des meilleures stratégies découvertes après l'évolution des AC, il est nécessaire de s'appuyer sur des outils formels garantissant une bonne interprétation des diagrammes espace-temps résultants. Cette analyse est une étape préliminaire dans la compréhension des mécanismes généraux par lesquels des calculs émergents sophistiqués peuvent être automatiquement produits. Crutchfield et Hanson [17], collègues de Mitchell au laboratoire de SFI, ont développé en

1993 un cadre théorique très puissant qui permet d'étudier les mécanismes des calculs intrinsèques intégrés dans les figures espace-temps en termes de *domaines*, de *particules*, et des *interactions de particules*.

Parfois, dans les diagrammes espace-temps il est possible que des régions dynamiquement homogènes apparaissent et seront évidentes à l'œil en tant que *domaines*, c'est-à-dire une région homogène dans laquelle le même motif [*pattern*] apparaît. Dans les autres cas, l'observation humaine est généralement insuffisante pour les détecter. Ainsi, la notion du domaine a été formalisée par l'adaptation de la théorie de calcul pour étudier la dynamique des AC. Les *particules* jouent le rôle d'un porteur de l'information tandis qu'avec *l'interaction* entre elles, des traitements de l'information s'effectuent. Cet outil formel a été largement utilisé dans l'étude du problème de la classification de densité et permettait d'aboutir à des résultats très encourageants.

Le problème de la classification de densité donne donc naissance à des problèmes qui font certes appel à l'informatique mais dont la résolution requiert des techniques purement liées à la théorie de calcul et à la théorie des systèmes dynamiques. D'autre part, la découverte des stratégies de calcul sophistiquées semble montrer qu'il existait une possibilité d'aller plus loin en ingénierie des systèmes multiprocesseurs décentralisés.

Parallèlement aux recherches qui se déroulaient sur la résolution du problème de la classification de densité, d'autres chercheurs se sont intéressés à un autre problème dit de la *synchronisation*. Il consiste à trouver un AC unidimensionnel à deux états, tel que, partant d'une configuration aléatoire, on arrive, après M itérations, à un motif dans lequel l'ensemble de ses cellules oscillent de manière synchrone en présentant toutes simultanément des 0 et des 1 à chaque pas de temps.

La tâche est non triviale car l'oscillation synchrone est une propriété globale d'une configuration, alors un AC à petit rayon (par exemple, $r = 3$) emploie seulement des interactions locales entre les cellules. Ainsi, comme la localité des interactions peut conduire directement à des régions de synchronisation locale, il est très difficile de concevoir un AC qui permettra d'assurer que les régions qui sont spatialement éloignées seront en phase. Et comme les régions qui ne sont pas en synchronisation peuvent être distribuées à travers tout le réseau cellulaire, un bon AC doit donc transférer de l'information sur une grande distance à travers le diagramme espace-temps ($\approx N$, N est la taille du réseau cellulaire) [19].

L'étude de ce problème peut bien couvrir de nombreux systèmes complexes naturels qui montrent l'existence du calcul émergent, Sipper écrit ainsi :

« The phenomenon of synchronous oscillations occurs in nature, a striking example of which is exhibited by fireflies. Thousands such creatures may flash on and off in unison, having started from totally uncoordinated flickerings. Each insect has its own rhythm, which changes only through local interactions with its neighbors' lights. Another interesting case involves pendulum clocks: when several of these are placed near each other, they soon become synchronized by tiny coupling forces transmitted through the air or by vibrations in the wall to which they are attached. » [24]

[Le phénomène des oscillations synchrones se produit dans la nature et un exemple frappant est de ce qui est manifesté par les lucioles. Des milliers de telles créatures peuvent s'allumer et s'éteindre à un unisson, en partant d'un scintillement totalement désordonné. Chaque insecte a son propre rythme, qui ne change que par l'intermédiaire d'interactions avec des lumières de ses voisins. Un autre cas intéressant est celui des pendules d'horloges. Quand plusieurs de ceux-ci sont placés à proximité les uns des autres, ils deviennent bientôt synchronisés par des petites forces transmises par la voie de l'air ou par les vibrations de la paroi à laquelle ils sont attachés.] [Ma TRD].

Ce problème a initialement été étudié en tant qu'une tâche non triviale par Das en 1995 au laboratoire de SFI [19]. Le but étant de mesurer la puissance de calcul des AC pour la résolution des problèmes nécessitant une coordination globale. Sipper aussi l'a étudié dans le cadre de ses études effectuées sur les AC non-uniformes. A l'instar du problème de la classification de densité, l'étude des diagrammes espace-temps résultants à partir de l'évolution des AC par des AE pour résoudre le problème de la synchronisation a pu confirmer l'utilité des stratégies découvertes précédemment pour l'ingénierie des calculs émergents. Il est à noter ici que ce problème n'a pas tardé de devenir très rapidement un concurrent important du problème de la classification de densité, et par conséquent il a largement été discuté par des chercheurs ayant évolué des AC par des AE.

4. La généralisation pour résoudre d'autres problèmes

Jusqu'ici, les recherches sur ce sujet avaient tendance à résoudre deux problèmes majeurs : celui de la classification de densité et celui de la synchronisation. Les résultats obtenus à partir de l'étude de ces deux problèmes ont fortement encouragé les chercheurs pour résoudre d'autres problèmes.

L'intérêt pour les ACEV va donc aller au-delà de ces deux problèmes et connaître par là un regain après l'apparition d'autres problèmes pouvant aussi s'imposer et conduisant à montrer la capacité des AC pour effectuer des calculs émergents nécessitant une coordination globale. Remarquons que dans cette période, les chercheurs ont presque quitté l'utilisation des AC unidimensionnels en s'orientant vers l'étude des AC bidimensionnels et même multidimensionnels afin de résoudre de nombreux problèmes. Le problème dit du « **damier** » [37] a par exemple été utilisé comme une tâche non triviale pour mesurer la puissance de calcul des AC. Il consiste à trouver un AC bidimensionnel à deux états, tel que, partant d'une configuration aléatoire, on arrive à un motif (une configuration) en damier après I itérations. Citons un autre problème, celui dit des « **bitmaps** » [37]. Il consiste à trouver un AC bidimensionnel à deux états, tel que s'il itère depuis une configuration initiale il aboutit à un état final désiré en moins de I itérations.

D'autres problèmes ont également été étudiés dans ce sens par des ACEV comme celui du **AND (OR et XOR)** [37] où partant d'une configuration dont toutes les cellules sont dans l'état de repos à l'exception de deux cellules positionnées au coins qui peuvent se trouver dans un état actif, l'automate doit arriver, après I itérations, à un motif unique ne contenant que des cellules actives si l'opération logique a été réalisée avec succès et à un motif dont les cellules se trouvent dans l'état de repos autrement.

Toutes ces recherches ne tendaient qu'à confirmer les insuffisances des techniques informatiques actuelles qui doivent être changées et remplacées par d'autres plus persuasives et sophistiquées notamment pour l'ingénierie des calculs dans les systèmes multiprocesseurs décentralisés. En effet, depuis quelques années, l'utilisation de l'outil informatique s'est développée de manière spectaculaire et les approches classiques de conception se heurtent à de nombreuses difficultés. Les applications informatiques qui sont de plus en plus complexes et distribuées peuvent donc être développées en profitant de la puissance sans cesse croissante des ordinateurs.

Petit à petit, l'utilisation des ACEV s'élargie vers la résolution d'autres variantes de problèmes. L'un des problèmes rencontrés par les communautés des chercheurs étudiant les AC est celui du problème inverse. Il s'agit de trouver un AC possédant des propriétés globales présélectionnées [33]. Effectivement, déduire des règles locales à partir d'un comportement global est extrêmement difficile en raison de l'absence d'un moyen décrivant le comportement d'un AC à partir de sa fonction de transition, d'ailleurs le problème de la classification des AC n'est pas encore résolu. A ce jour de nombreuses tentatives ont été émises, néanmoins aucune d'entre elles ne s'est imposée de façon définitive et le problème de la classification reste encore ouvert. Les chercheurs doivent alors vaincre l'ampleur d'un espace de recherche énorme. Une tâche qui semble être très difficile car semblable à trouver une aiguille dans un tas de foin. La meilleure solution à cette impasse est qu'une stratégie d'exploration doit être mise à profit pour chercher l'AC souhaitable.

L'un des chercheurs qui ont parvenu à obtenir des résultats très prometteurs fut Emmanuel Sapin. Ses travaux portent sur l'application des ACEV pour la découverte des AC universels. L'universalité est une propriété fondamentale dans le monde des AC du fait qu'elle peut jouer un rôle primordial pouvant peut-être conduire à proposer une classification pour les AC, Sapin écrit donc :

" La découverte d'un grand nombre d'automates universels peut permettre d'élaborer une nouvelle classification des AC, plusieurs classifications existantes étant déjà basées sur les notions d'universalité et de complexité (Wolfram, 1984) (Gutowick et al, 1988). D'autre part la connaissance d'un nombre important d'automates universels pourrait permettre de comparer les règles de ces automates pour chercher des similitudes entre elles et caractériser les paramètres, s'ils y existent, permettant l'émergence de comportements complexes globaux tels que l'universalité " [36]

Ici la capacité de calcul se traduit par la possibilité d'effectuer la calculabilité universelle où d'après la bonne configuration initiale, l'AC sera capable de simuler un ordinateur programmable, avec des portes logiques, dispositifs de synchronisation, et ainsi de suite. Jusque les années 90, le jeu de la vie restait le seul AC à deux états ayant la propriété d'universalité, Sapin s'est alors demandé s'il est possible de découvrir d'autres qui peuvent l'avoir aussi. L'universalité de *Life* a été prouvée par la réalisation des portes logiques telles

que ET, OU, NON et d'autres propriétés de mémoire en lecture / écriture par des interactions entre figures stables où chaque nombre est codé par un faisceau de planeurs, unité de base qui sert à la circulation de l'information, généré par un lance-planeurs; on peut donc dire que le planeur est l'équivalent d'un bit en théorie de l'information.

Sapin avait procédé par analogie [35]. Partant d'une démarche cartésienne en appliquant le principe de Descartes dont l'énoncé est : " diviser pour régner ", il a étudié le problème le décomposant en problèmes plus petits : chercher d'abord des AC acceptant des planeurs puis trouver ceux acceptant un lance-planeurs, à chaque étape un processus évolutionnaire a été utilisé afin d'extraire l'AC cherché. Finalement, il a aboutit à découvrir un AC à deux états possédant la propriété d'universalité dans le sens logique où son automate était capable de simuler des portes logiques, comme dans *Life*, mais aussi dans le sens d'universalité de Turing où il était capable de simuler *Life* lui-même.

L'originalité de son travail tient à ce qu'il est le premier à avoir conduit une découverte d'un AC universel à deux états autre que *Life* appartenant à l'espace I défini par Wolfram dans [9] et ayant la particularité suivante :

Tous les AC de cet espace sont à deux états, s'exécutant sur des grilles à deux dimensions, isotropiques et chaque cellule prend en compte ses huit voisines pour constituer son voisinage. En effet, Sapin a relancé par ses études le débat concernant la question N°16 posée par Wolfram en [10], et dont l'énoncé est : " *How common are computational universality and undecidability in cellular automata* " en donnant quelques éléments de réponse à cette question, même si de façon partielle.

En 2005, Sapin propose son article [36], c'était le fruit d'une série de travaux continus et commencés depuis sa préparation pour l'obtention de son doctorat. Il a proposé un cadre théorique concrétisé sous forme d'une approche basée sur les AE pour la recherche des AC universels qui n'était en réalité que la généralisation de l'approche utilisée lors de la découverte de son AC universel.

En parallèle, d'autres chercheurs se sont intéressés par une autre notion (type) de calcul : la réalisation des tâches particulières. Dans ce cas, la fonction de transition est interprétée comme un programme traitant une configuration initiale interprétée comme son entrée, l'AC itère pendant un certain nombre de pas de temps ou jusqu'à ce qu'il atteigne certains motifs

objectifs, éventuellement, un point fixe. La configuration finale est interprétée comme sa sortie.

Contrairement aux travaux discutés précédemment, ici les configurations initiales ont généralement la particularité d'être connues au préalable et il ne sera donc demandé que de trouver l'AC souhaitable à la réalisation de la tâche en question. Remarquant que l'étude des AC dans cette période tendait vers l'invasion de plusieurs domaines aussi variés que l'imagerie, la reconnaissance des formes, la cryptographie, l'étude du développement des systèmes urbains, etc., les chercheurs trouvaient donc que les ACEV constituent un outil très puissant pour vaincre cette variété de types de problèmes et par conséquent de déduire les règles locales ayant la capacité de résoudre de tels problèmes. Cet intérêt croissant pour ce type de calcul par les chercheurs ne peut s'expliquer que par sa généralité en raison de sa possibilité de porter sur plusieurs classes de problèmes. En tout état de cause, il paraît clair que nous ne pouvons aborder ce très grand nombre de travaux et par conséquent nous sommes limités à sélectionner un domaine d'application paraissant important pour en discuter : celui de l'application des AC pour résoudre des problèmes liés aux traitements d'images.

Le domaine du traitement d'images a été largement étudié dans ce contexte. La possibilité d'appliquer des AC pour effectuer des tâches de traitement d'images se pose comme une conséquence naturelle de leur architecture. Pour les AC à deux dimensions, une cellule (ou un groupe de cellules) peut correspondre à un pixel d'une image, et avec la dynamique de l'AC conçue la tâche du traitement d'images désirée s'effectue [24]. De telles tâches sont généralement incluses dans le cadre de la résolution des problèmes par émergence (*cf. II.1.5*) qui consiste à découvrir des règles d'interactions locales, qui seront en mesure de produire une solution globale au problème. Cependant, la largeur de l'espace de recherche à laquelle on fait face entrave la déduction des règles pouvant réaliser la tâche demandée. Une solution possible est l'automatisation du processus de la recherche des règles adéquates, et cela grâce à un processus évolutionnaire pour *l'apprentissage* des AC.

Jusqu'à présent, il n'existe pas de définition générale, universellement acceptée pour le concept d'apprentissage, car il touche à trop de notions distinctes et dépendantes du contexte où il est utilisé. Généralement, pour un AC cette notion se réfère à l'acquisition de savoir-faire en modifiant ses règles de transitions de façon à lui permettre une amélioration du rendement. Le processus d'apprentissage peut également se faire par une approche évolutionnaire. De nombreux travaux ont été menés pour étudier la possibilité de faire apprendre des AC afin

d'effectuer plusieurs tâches de traitement d'images telles que : le cryptage / décryptage d'images [39], la segmentation d'images [18], [34], [40], le filtrage d'images [39]. La puissance des algorithmes d'apprentissages dans ce sens est que tout ce qui est requis est [39]:

- Un ensemble d'images pour l'apprentissage.
- Un ensemble d'images cibles correspondantes (c'est-à-dire, idéales).
- Une fonction objective d'évaluation de la qualité des images actuellement produites par l'AC, c'est-à-dire, en déterminant l'erreur entre l'image cible et celle produite par l'AC comme sortie.

Un excellent exemple dans ce sens est construit au sein du laboratoire LIRE de notre université par le chercheur Mohamed Batouche [40], ancien professeur à l'université Mentouri, où la tâche de la segmentation des images binaires a été résolue par émergence en évoluant des populations d'AC par un AG afin de trouver des règles locales garantissant la résolution de cette tâche en se basant sur la notion d'apprentissage. Des résultats très prometteurs ont été obtenus en montrant par là la puissance de calcul des AC pour la résolution des problèmes liés au domaine du traitement d'images.

5. Conclusion

En conclusion, l'histoire des ACEV peut donc être divisée en trois grandes périodes. La première période (1988 à 1995) fut principalement tournée vers l'apparition du problème de la classification de densité comme une tâche complexe pouvant mesurer la puissance de calcul des AC. Une approche évolutionnaire a été introduite pour explorer les espaces d'AC afin de découvrir des stratégies sophistiquées de calcul émergent. Quelques unes des stratégies calculatoires ont par conséquent été découvertes ouvrant par là la voie à une nouvelle approche de se développer. La seconde période (1995 à 2002) vit le développement des études liées au problème de la classification de densité, et il y a là une diversité de travaux visant soit à résoudre ce problème définitivement, soit à l'utiliser comme une mesure de puissance d'autres approches évolutionnaires. A partir de l'an 2002, les ACEV furent étudiés comme un outil puissant pour la réalisation de quelques tâches qui n'étaient pas encore résolues. En effet, à cette période l'étude des AC se dirige vers une diversité d'applications touchant à de nombreux domaines. Les chercheurs ont donc constaté que les ACEV peuvent vaincre cette variété sachant que la difficulté de la conception des AC a largement limité leurs applications.

Il semble donc qu'une nouvelle approche basée sur l'évolution des AC se soit développée. Ses caractéristiques principales étant de modéliser des AC par des chromosomes puis les évoluer selon une stratégie d'évolution et enfin déduire des AC ayant des particularités calculatoires.

Le chapitre suivant est donc consacré à l'étude formelle et technique de ces caractéristiques.

*Les automates
cellulaires
évolutionnaires :
Etudes
techniques et
formelles*

*Chapitre
2*

Introduction

Partie I : Automates cellulaires

6. Origines et développement
7. Définitions de base
8. Terminologie
9. Universalités des AC
10. Le phénomène d'émergence dans l'univers des AC

Partie II : Algorithmes génétiques

1. Source d'inspiration
2. Algorithmes génétiques
3. Terminologie
4. Comment ça fonctionne ?

Partie III : Vers des automates cellulaires évolutionnaires

1. Automates cellulaires évolutionnaires
2. D'un automate cellulaire vers un chromosome
3. Construction d'un automate cellulaire évolutionnaire
4. Etude de cas

Conclusion

Les automates cellulaires évolutionnaires :

Etudes techniques et formelles

Introduction

Le chapitre qui suit a pour but de présenter un nouveau concept appelé ‘automate cellulaire évolutionnaire’. Par AC, on entend une simulation d'un phénomène, d'un comportement dynamique, d'un calcul émergent et par AE on entend une simulation de l'évolution, d'une optimisation d'une solution, d'un outil d'exploration. Comment deux concepts aussi différents peuvent-ils s'associer pour désigner un nouvel objet ?

Nous avons choisi d'aborder la question en présentant séparément chacun des concepts et c'est pourquoi permettre ainsi de cerner le mode de fonctionnement de ce nouvel objet.

Dans la première partie, nous allons exposer la classe du modèle abstrait d'AC qui constitue un cadre théorique très puissant pour l'étude des systèmes complexes. Nous nous intéresserons dans la seconde partie à l'étude des AG qui constituent un exemple représentatif d'un ensemble de méthodes connues sous le nom d'AE. Enfin, nous aborderons dans la troisième partie le concept métis à partir du mixage des deux premiers, celui des ACEV.

Nous avons essentiellement étudié ces concepts d'un point de vue formel et technique. En effet, en tant qu'informaticiens, ces deux aspects soulèvent une grande partie de nos préoccupations, car le premier sert à assurer la sûreté des bases théoriques selon lesquelles sont construites de nombreuses entités informatiques alors que le second sert à assurer la faisabilité et donc la possibilité d'exploitation par les machines.

Partie I : Automates cellulaires

"... One might have thought that if a program was simple it should only do simple things. But amazingly enough, that isn't even close to correct. And in fact what I've discovered is that some of the very simplest imaginable computer programs can do things as complex as anything in our whole universe. It's this point that seems to be the secret that's used all over nature to produce the complex and intricate things we see..."

[... On aurait pu croire que si un programme était simple il ne devrait faire que des choses simples. Mais assez étonnamment, ce n'est même pas près de vrai. Et en fait ce que j'ai découvert que certains des très simples programmes d'ordinateurs imaginables peuvent faire des choses aussi complexes que quoi que ce soit dans l'ensemble de notre univers. C'est ce point qui semble être le secret qui est utilisé dans la nature pour produire la complexité et les choses complexes que nous voyons...] [Ma TRD].

C'était un extrait d'une discussion avec Stephen Wolfram, à propos du sujet de son livre " A new kind of science "², des règles extrêmement simples pouvaient alors émerger des structures beaucoup plus complexes. Une remarque qui semble être d'une grande importance et donc mérite d'avoir un intérêt pour en discuter. En quoi consisterait un AC ? Quelles sont les définitions de base d'une telle théorie ? Que peut-il faire ?

1. Origines et développement

Dans les années quarante, le chercheur Von Neumann, père des architectures des ordinateurs actuels, a étudié les automates autoreproducteurs, le but fut de savoir s'il était possible de concevoir une machine capable de s'auto reproduire, c'est à dire produire une copie d'elle-même. En effet, ce mécanisme correspond à l'interprétation actuelle du fonctionnement de la molécule d'ADN découverte plus tard. Etant donné que l'ADN n'était pas encore découvert au cours de cette période, Von Neumann souhaite alors mieux comprendre la logique qui sous-tend de tels phénomènes.

Von Neumann trouva son chemin pour résoudre le problème posé grâce aux travaux du mathématicien Stanislas Ulam, son collègue au laboratoire de Los Alamos. Ce dernier a

² Source : wolframscience.com

étudié la génération des structures graphiques définies de façon récursive. La source étant des règles simples permettant d'engendrer des figures complexes et esthétiques pouvant par fois se répliquer. Il a utilisé un espace à deux dimensions divisé en « cellules ». Chacune d'entre elles pouvait avoir deux états : allumé ou éteint. Partant d'une configuration donnée, la génération suivante était déterminée en fonction des règles de voisinage. Il suggéra alors à Von Neumann d'utiliser ce qu'il appelait les « espaces cellulaires » pour pallier les difficultés pratiques qui se posaient lors de la construction de l'automate autoreproducteur [32] [49].

Von Neumann a, tout de suite, construit son automate autoreproducteur, c'était un automate bidimensionnel à 29 états avec un voisinage de 5 cellules (cellule cible + 4 cellules voisines contiguës). Il a également décrit comment un constructeur universel pouvait être construit, c'est à dire une structure capable de générer n'importe quelle configuration stockée dans une structure de stockage [54].

Ce n'est qu'en 1966 que le terme *automate cellulaire* est créé par Arthur Burks en coïncidence avec la publication du premier grand ouvrage consacré au problème de l'autoreproduction : *Theory of self-reproducing automata*. Rappelons que dans les années soixante, quelques-uns des problèmes mathématiques ont été résolus par des AC comme celui de « **la synchronisation des fusiliers** » par exemple [32].

L'application des AC n'est élargie qu'après l'apparition du désormais fameux *Jeu de la Vie* (*Life*) de John Horton Conway en 1970. Nous n'allons pas l'aborder ici car il le sera par la suite. Beaucoup de chercheurs se sont intéressés à son fonctionnement plus tard afin de maîtriser les énigmes qui sont nées de son étude. Parallèlement aux recherches qui se déroulaient sur *Life*, d'autres chercheurs étudiaient d'autres AC ayant des règles simples et qui conduisaient aussi à des comportements imprévisibles. Plus tard, dans les années 80, les chercheurs se sont également dirigés vers la physique où les AC ont été utilisés comme modèles de simulation des comportements des molécules et d'autres phénomènes physiques. Depuis le début des années 90, l'utilisation des AC s'est généralisée en touchant à de nombreux domaines, notamment la cryptographie, l'imagerie, l'étude du développement de systèmes urbains ... etc.

2. Définitions de base

Un AC se définit à l'aide de deux types de caractéristiques : **structurelles** et **fonctionnelles** [32]. Les premières concernent **l'aspect topologique** du réseau cellulaire, les secondes concernent **l'aspect dynamique** de l'évolution du réseau au cours du temps. En règle générale, le réseau des cellules peut prendre corps dans des espaces à D dimensions, D étant une, deux ou trois dimensions ou encore plus. Théoriquement, il n'y a pas de limite à la dimension d'un automate, si ce n'est la puissance de calcul des machines sensées le reproduire. Par exemple, dans le cas d'une matrice à $2D$, la structure du réseau cellulaire peut-être de deux types :

- **La structure en île** : Les cellules du réseau sont virtuellement entourées par des cellules mortes.
- **La structure en tore** : Les cellules du bord haut de la matrice peuvent contacter celles du bord bas, et de même pour les cellules des bords gauche et droit.

Le réseau cellulaire peut également être variable comme, par exemple, le réseau orthogonal ou hexagonal.

Dans un AC une cellule n'a conscience que d'un sous-ensemble de cellules dites *voisines* ayant une influence sur elle. On dit qu'un voisinage d'une cellule est aveugle si chacune des cellules le constituant joue un rôle identique, la cellule concernée n'aura donc connaissance que des différents états des cellules voisines sans savoir quel état correspond à quelle cellule. Ce type de voisinage est le plus souvent utilisé par la plupart des AC sans le préciser [54].

Le fonctionnement de chaque cellule du réseau peut être caractérisé par l'*automate fini* (V, v_0, f) [32] :

- V est l'ensemble des états cellulaires.
- v_0 un état particulier appelé état de repos.
- f est la *fonction de transition* qui à chaque voisinage qui est n -tuple d'éléments de V associe un élément de V .

V étant l'ensemble des états que peut prendre une cellule. Le plus souvent limité à 2, mais il n'y a aucune limite à ce nombre, c'est le cas par exemple pour l'automate de Von Neumann qui était à 29 états. Dans la pratique, les états cellulaires sont souvent représentés par des couleurs ce qui permet de suivre l'évolution de l'automate. Comment l'évolution de tel automate aura lieu ?

La fonction de transition f est un ensemble de règles qui permettent de déterminer le nouvel état d'une cellule en fonction de son état précédent et de l'état précédent de son voisinage. Généralement, ces règles ne sont pas explicitées, mais résumées sous la forme de métarègles de type « *si...alors* ».

L'automate évolue discrètement par génération où le temps s'écoule par à-coups. Ceci signifie qu'à la génération t , chaque cellule décide d'après sa fonction de transition que devient son état au futur c'est-à-dire au temps $t+1$. Une fois chaque cellule a déterminé son prochain état, et seulement à ce moment là, elle passe au nouvel état calculé d'une façon à ce que le réseau cellulaire sera entièrement remis à jour de manière synchrone. On parle alors d'une simulation d'un traitement parallèle. Formellement, on peut exprimer l'évolution de l'automate par la formule 2.1.

$$S_i(t+1) = f(\{S_j(t)\}) \quad \text{où } j \in V_i \wedge V_i \text{ désigne le voisinage de la cellule } i$$

Formule 2.1. Règle de mise à jour d'un AC

Un AC pouvait donc être défini par un aspect topologique décrivant la façon par laquelle les cellules sont arrangées sur le réseau cellulaire et un autre fonctionnel caractérisé par le voisinage, l'ensemble d'états et la fonction de transition qui décrit son évolution.

3. Terminologie

Nous avons choisi quelques concepts clé que se partagent la plupart des AC [54].

- **Voisinage :** Pour chaque cellule le passage d'un état à un autre est déterminé en examinant les états des cellules voisines. Seules les interactions locales sont permises entre les cellules.
- **Parallélisme :** Toutes les cellules constituant le réseau cellulaire de l'automate sont mises à jour de manière simultanée et synchrone.

- **Déterminisme** : Un AC est dit déterministe si sa fonction de transition est une fonction classique c'est-à-dire une même entrée donne lieu toujours à la même sortie, donc la donnée des états des cellules voisines détermine à elle seule le nouvel état d'une cellule. Certains AC dits stochastiques introduisent un facteur de probabilité dans leurs fonctions de transitions en faisant intervenir des variables aléatoires, donc une même entrée peut donner lieu à plusieurs sorties différentes, en d'autre terme l'évolution d'une configuration par une même fonction de transition n'est pas toujours la même.

- **Homogénéité** : On dit qu'un AC est homogène s'il satisfait les conditions suivantes :
 - La topologie du réseau cellulaire est régulière.
 - La fonction qui calcule le voisinage doit être uniforme pour toutes les cellules.
 - L'évolution de l'automate se définit par une seule règle de transition qui s'applique à toutes les cellules.

- **Discrétisation** : Un AC s'évolue dans le temps de manière discrète, c'est-à-dire génération après génération, ce qui s'oppose avec nombreux phénomènes physiques continus.

4. Universalités des AC

Dans la partie précédente, nous avons évoqué la définition d'un AC ainsi que la définition de quelques concepts qui lui sont liés. Nous allons maintenant étudier la propriété d'universalité dans l'univers des AC. D'une part, cette notion est fondée sur un principe très simple : " *la capacité de faire* ", et d'autre part, les AC ont toujours été considérés en tant *des systèmes dynamiques* que *des systèmes de calcul*. Par projection, nous arriverons à construire deux types différents d'universalités : la première porte sur la capacité de calcul dite *universalité Turing* tandis que la seconde porte sur la capacité de produire certains comportements dynamiques dite *universalité intrinsèque*. Dans tous les cas, la notion d'universalité dans le contexte des AC vise sous une forme ou sous une autre à répondre à une question primordiale: y a-t-il un AC qui *peut faire tous* ?

4.1. Universalité Turing

4.1.1. Machine de Turing vs automate cellulaire

Selon la thèse Church / Turing tout problème de calcul bien formé basé sur une procédure algorithmique peut être résolu par une machine de Turing. On peut se demander de savoir jusqu'à quelle mesure peuvent les AC s'imposer. Nous pouvons ainsi faire une simple comparaison entre un AC et une machine de Turing [2]. Prenons l'exemple figurant dans [32], soit la machine :

- états : $\{E1, E2, E3\}$; symboles : $\{X, A\}$; état initial : $E1$
- tableau de marche (symbole à écrire, déplacement droite / gauche, nouvel état)

symbole lu / état	$E1$	$E2$	$E3$
X	$(X, D, E1)$	$(A, G, E3)$	$(X, D, E1)$
A	$(X, D, E2)$	$(A, D, E2)$	$(A, G, E3)$

Table 2.1. Table de marche de la machine de Turing étudiée

Remarquons que pour toute suite de A accolés, cette machine déplace une telle séquence d'une case vers la droite. L'observateur voit les choses comme si les séquences de A se déplacent vers la droite. Si on désire opérer cela pour toutes les séquences initialement contenues dans le ruban et de façon **simultanée**. Il est clair qu'une machine de Turing ne peut le faire : les séquences doivent être déplacées séquentiellement les unes après les autres. Cela est très facile avec un AC unidimensionnel, il suffit de considérer l'automate suivant :

- Si une cellule possède un état X à sa gauche, elle devient X .
- Si une cellule possède un état A à sa gauche, elle devient A .

Les AC permettent donc de faire évoluer des configurations de façon **parallèle**, ce qui est carrément impossible avec une machine de Turing.

4.1.2 L'universalité Turing des AC

Von Neumann, puis Codd ont prouvé de façon constructive l'équivalence formelle de la puissance de calcul des AC avec les machines de Turing. Cela a été fait en démontrant qu'à toute machine de Turing universelle il existe un AC la simulant. La difficulté principale

pour établir une telle démonstration est la nature *parallèle* des AC : le fonctionnement d'un AC est *parallèle* où à chaque pas de temps toutes les cellules peuvent changer d'état alors que celui d'une machine de Turing est *séquentiel* où à chaque pas de temps seule une case du ruban a la possibilité d'être modifiée. Il est donc impératif de stopper le parallélisme pour en aboutir. Ceci conduit généralement à utiliser un grand nombre d'états différents, limitant par là l'intérêt pour lequel les AC ont été inventés. Formellement la correspondance entre un AC et une machine de Turing peut s'établir de la façon suivante [43] :

Soit Z une machine de Turing. De manière formelle, Z se définit par la donnée des quatre éléments suivants : $Z = (V, Q, q_0, P)$ où

V est le vocabulaire du ruban.

Q est un ensemble des états de la machine Z .

q_0 est l'état initial ($q_0 \in Q$).

P est un ensemble de quintuples définis sur : $Q \times V \times V \times \{D, G, N, H\} \times Q$.

Soit QT un quintuple défini par $QT = (q, s_i, s_j, \alpha, q')$, son interprétation se fait de la manière suivante :

Si la machine Z est dans l'état q **et** le symbole visé par la tête de L / E est de s_i alors

- s_i est remplacé par s_j .
- Déplacement de la tête de L / E d'une case selon la valeur de α .
- La machine Z passe à l'état q' .

On peut maintenant définir un AC A qui soit capable de simuler la machine Z . Le réseau cellulaire de l'automate A est un tableau unidimensionnel de cellules. L'ensemble des états cellulaires V' est défini sur le produit cartésien $V \times (Q \cup \{p\})$, p est un état supplémentaire indiquant l'absence de la tête de L / E à une case donnée. Les états cellulaires de A ont donc la forme (v, q) où v correspond au contenu du ruban et q correspond à l'absence ou à la présence de la tête de L / E à cette cellule. Chaque cellule ne peut contacter que celles situées juste à sa droite et à sa gauche, donc trois cellules au total par voisinage. Le fonctionnement de chaque cellule est déterminé par la fonction de transition f définie comme suit :

Chaque cellule examine son état et les états de ses voisins en vérifiant la présence de la tête de L / E puis elle décide que devient son état futur, plusieurs cas peuvent se figurer.

- La tête est positionnée sur la cellule elle-même c'est-à-dire la deuxième composante de son état est différente de p , dans ce cas la cellule doit mettre à jour son état immédiatement en remplaçant sa première composante par la valeur du symbole se trouvant dans le quintuple de la machine Z correspondant et sa deuxième composante par p s'il y a un déplacement.
- La tête est positionnée sur la cellule se trouvant à sa gauche, si le quintuple indique un déplacement vers la droite alors la cellule doit mettre à jour la partie droite de son état en remplaçant la valeur p par la valeur de l'état se trouvant dans ce quintuple.
- Le même processus est répété pour la cellule se trouvant à sa droite.
- Dans les autres cas l'état de la cellule reste inchangé.

4.2. L'universalité intrinsèque des AC

Un AC est considéré comme intrinsèquement universel si et seulement s'il est capable de **simuler pas à pas le comportement** de n'importe quel autre AC de même nature topologique (dimension). Cette notion a été introduite pour la première fois par J. Albert [11]. L'idée repose sur le principe suivant [43] :

- Chaque cellule de l'automate simulé est équivalente à un groupe de cellules identiques et voisines dans l'automate simulateur, l'ensemble de tous les groupes doit régulièrement recouvrir le réseau des cellules.
- A chaque état de l'automate simulé correspond une configuration d'états de groupe dans l'automate simulateur.
- Une étape de l'automate simulé se réalise en plusieurs étapes exécutées par l'automate simulateur.

On doit également garder à l'esprit que la notion d'universalité intrinsèque est **plus forte** que l'universalité Turing, car un automate qui est capable de simuler **tous** les autres automates peut forcément simuler un automate universel dans le sens de Turing mais le contraire n'est pas toujours vrai.

5. Le phénomène d'émergence dans l'univers des AC

La propriété d'émergence est généralement illustrée par la formule " *le tout est plus que la somme des parties* ". Elle présuppose une apparition de nouveauté (propriétés, formes, structures, fonctions) mais pourtant elle implique qu'on ne peut décrire, d'expliquer ou de prédire ces nouveaux phénomènes en termes physiques à partir des conditions de base définies aux niveaux inférieurs [27]. Les AC constituent un outil précieux d'analyse de l'émergence du fait qu'à partir de simples interactions entre les cellules il est possible d'exhiber des comportements très complexes, nous allons étudier cette propriété dans le cas du fameux *Jeu de la Vie* [4].

John Horton Conway est un mathématicien anglais qui s'est intéressé par les travaux de Von Neumann concernant les AC, son but était d'inventer un automate qui soit plus simple que celui de Neumann mais qui a la propriété d'universalité au sens de Turing. Effectivement, c'est ce qui a été fait, après deux ans de travail continu il a abouti à ce que l'on connaît maintenant sous le nom de *Game Of Life* ou encore *Life* en abrégé.

Les règles du *Jeu de la Vie* sont extrêmement simples. Les cellules peuvent se trouver dans deux états qui sont : **vivant** / **mort**. Au départ, toutes les cellules se trouvent dans l'état mort sauf pour un petit nombre d'entre elles. L'espace cellulaire est composé de cellules arrangées en une matrice (grille) où chaque cellule ne peut contacter que celles se trouvant dans les huit cases adjacentes à elle.

On note par N_v le nombre de cellules vivantes voisines à une cellule donnée. L'évolution de l'automate est déterminée par la fonction de transition suivante [32] :

- Une cellule vivante meurt si $N_v = 1$: cela correspond à un état d'isolement de cellule.
- Une cellule vivante meurt si $N_v \geq 4$: cela correspond à un état de surpeuplement autour de la cellule.
- Une cellule morte peut devenir vivante si $N_v = 3$: cela correspond à une reproduction « trisexuée ».

Ce n'est que quelques dizaines d'itérations, la population des cellules vivantes diminue sensiblement. L'écran se peuple alors des structures étranges, des motifs géométriques qui oscillent et changent de forme [45]. Généralement, les chercheurs classent les objets selon leur forme de stabilité. Les objets les plus simples sont ceux qui restent identiques à eux-mêmes au cours de l'évolution. Viennent ensuite les objets qui dont l'évolution est périodique, nommés *oscillateurs*. Une autre classe d'objets importants est celle des objets périodiques qui se translatent avec le temps. On peut s'interroger : comment cette complexité peut-elle provenir d'une règle aussi simple ? [32]

Voici quelques structures très simples que l'on peut observer dans Life.

- **structures stables :**

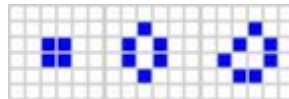


Figure 2.1. Structure stable

- **L'oscillateur du clignotant :** C'est l'oscillateur le plus simple, il est constitué de trois cellules alignées, et possède une période de 2.

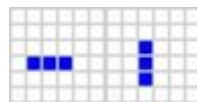


Figure 2.2. Oscillateur clignotant

- **Les planeurs :** Le planeur est l'exemple le plus simple d'un objet translatant dans le temps. Il apparaît de façon spontanée lors de l'évolution. Il est capable, après un certain nombre de pas, de produire une copie de lui-même décalée par rapport à son endroit initial.

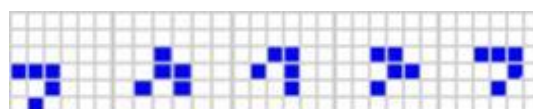


Figure 2.3. Planeur de cinq cellules

En conclusion, on peut dire que nous pouvons maintenant comprendre pourquoi des chercheurs comme Wolfram et Langton se sont intéressés par l'étude du modèle abstrait d'AC, cet intérêt ne peut être justifié que par la formule : " *puissance + simplicité* ", leur puissance se reflète par la capacité d'exhiber tous les types de comportements dynamiques (périodique, point fixe, chaotique et complexe) mais aussi par la capacité de réaliser toute sorte de calcul (universalité), en revanche leur simplicité se manifeste par des règles très simples contrôlant l'évolution de tels automates mais conduisant à un comportement global énorme.

Partie II : Algorithmes génétiques

" J'espère prouver que la nature possède les moyens et les facultés qui lui sont nécessaires pour produire elle-même ce que nous admirons en elle. "

Par ces mots a exprimé le naturaliste français Jean Baptiste de Monet, chevalier de LAMARCK ses croyances envers la nature, à notre tour nous sommes désormais, en tant qu'informaticiens, tentés de dire :

" Espérons prouver que la nature possède les moyens et les facultés qui sont nécessaires à la machine pour produire ce que nous admirons en elle. "

En effet, la biologie a souvent été une source d'inspiration dans plusieurs domaines, notamment dans le domaine de l'informatique. L'une des techniques les plus marquantes était celle des algorithmes génétiques (AG) qui inspirent des mécanismes de la sélection naturelle élaborés par Charles Darwin et de la génétique moderne. En quoi consisterait un AG ? Quelles sont les définitions de base d'une telle discipline ? Que peut-il faire ?

1. Source d'inspiration

En 1859 Charles Darwin expose sa théorie de l'évolution des espèces, il a essayé de montrer que l'apparition d'espèces distinctes se fait par le biais de la sélection naturelle des *variations* individuelles [1]. D'après cette théorie, la sélection naturelle est fondée sur la notion de *la lutte pour vivre* due à une population qui tend naturellement à s'étendre sachant qu'elle dispose d'un espace et de ressources finis. Il en résulte que les individus les *plus adaptés* tendent à survivre plus longtemps et à se reproduire plus aisément. Les êtres vivants se sont donc, sous l'influence des contraintes extérieures, graduellement adaptés à leur milieu naturel à travers le processus de reproduction.

Darwin a supposé que ces variations individuelles apparaissent au hasard ce qui permet d'expliquer les phénomènes d'évolution et d'adaptation sans avoir recourt ni à une création, ni à une modification directe de l'hérédité par le milieu, ni même à une finalité. Le terme « *adapté* » se réfère à l'environnement, que l'on peut définir comme étant l'ensemble des conditions externes à un individu, ce qui inclut les autres individus. Les lois de variations (croisements et mutations) furent expliquées sept ans à peu près plus tard par Mendel (en

1866) après la publication de l'article retraçant dix années d'expériences d'hybridation chez les végétaux (recombinaison des gènes) en proposant un mécanisme pour l'hérédité, c'est-à-dire la transmission des caractères d'un individu à ses descendants [53].

2. Algorithmes génétiques

Les AG sont apparus pour la première fois en 1975 par J. Holland et ses collègues à l'université de Michigan [6]. Ils appartiennent à la famille des algorithmes évolutionnaires (AE) qui reposent sur le principe de compétition entre individus d'une population : les individus qui sont mieux adaptés aux contraintes survivent alors les autres périssent en laissant par conséquent leurs places à d'autres nouveaux formés par la descendance des premiers.

Historiquement, trois variantes d'AE ont isolément été développées par trois grandes écoles :

- La programmation évolutionnaire [3].
- Les stratégies d'évolution [5].
- **Les algorithmes génétiques [6].**

Plus tard, une quatrième classe a vu le jour : la programmation génétique (GP) proposée par J. Koza en 1988 qui était auparavant un sous groupe des AG. Ces quatre classes d'algorithmes ne se différencient principalement que sur le détail d'implantation des opérateurs et des procédures de sélection et remplacement dans la population.

En outre, les AG s'appuient sur des techniques dérivées de la génétique moderne et c'est pourquoi on dit qu'ils sont fondés sur le *Néo-Darwinisme* (c'est-à-dire inspirant de la théorie de l'évolution et de la génétique moderne). Nous parlerons donc *d'individus, de population, de gènes, de chromosomes, de parents, de descendants, de reproduction, de croisement, de mutations, etc.* Un vocabulaire qui semble être directement calqué sur celui des deux théories adoptées.

Initialement, deux buts principaux pour lesquels ont été inventés les AG :

- Mettre en évidence et expliquer rigoureusement les processus d'adaptation des systèmes naturels.
- Concevoir des systèmes artificiels possédant les propriétés des systèmes naturels.

3. Terminologie

Avant d'aborder comment un problème soit résolu à l'aide d'un AG, nous devons également définir le vocabulaire employé. On rappelle ici que les sources d'informations consacrées au sujet se trouvent disséminées dans de nombreux articles et revues le plus souvent accessibles via un service payant ce qui nous a obligés de compter sur différents sites d'Internet et on ne peut donc garantir une meilleure couverture des informations exposées. Les sources, qui paraissaient intéressantes et pertinentes, qu'on a utilisées sont : ([43], [53], [50], [29]).

- **Chromosome** : En biologie, il est défini comme le porteur de l'information génétique nécessaire à la construction et au fonctionnement d'un organisme. Dans le cadre des AG, il correspond à un élément représentant une solution possible d'un problème donné.
- **Gène** : En biologie, il représente une partie du chromosome, chaque chromosome est constitué d'un certain nombre de gènes. Pour un AG, chaque chromosome est divisé en un ensemble d'unités le constituant dites gènes.
- **Génotype** : Dans les systèmes naturels, l'ensemble du matériel génétique est appelé le génotype. Dans les AG, l'ensemble des chaînes est appelé structure.
- **Phénotype** : Dans les systèmes naturels, l'organisme formé par l'interaction de l'ensemble du matériel génétique avec son environnement est appelé le phénotype. Dans les AG, les structures décodées forment un ensemble de paramètres donné, ou une solution ou un point dans l'espace des solutions.
- **Allèle** : Dans les systèmes naturels, l'allèle est une composante du gène. Les allèles sont les différentes valeurs que peuvent prendre les gènes. Dans les AG, l'allèle est également appelé valeur caractéristique.
- **Locus** : Le locus est la position d'un gène dans le chromosome.
- **Individu** : En biologie un individu est une forme qui est le produit de l'activité des gènes. Pour un AG, il est réduit à un chromosome et on l'appelle donc chromosome ou individu pour désigner un même objet.

- **Population** : Dans un système naturel, une population est simplement un ensemble d'individus. Par analogie, elle se définit comme l'ensemble des chromosomes. Elle est aussi appelée une génération.
- **Parents** : Dans un système naturel, les individus peuvent se reproduire en créant de nouveaux individus formant une nouvelle génération afin d'assurer la continuité de la vie. Dans le cadre d'un AG, les parents correspondent aux individus pouvant s'imposer ce qui donne naissance à de nouveaux individus descendants afin de former une nouvelle génération.
- **Descendants** : Dans un système naturel, la continuité de la vie se fait par la reproduction des individus de la population, il en résulte la naissance d'un certain nombre d'individus dits descendants et qui participent à la création de la nouvelle génération. Dans un AG, les descendants peuvent être considérés comme étant le résultat direct du processus de la reproduction des parents, chaque descendant hérite des caractéristiques issues de ses parents.

Les autres concepts comme le croisement et la mutation seront discutés dans une section suivante.

4. Comment ça fonctionne ?

Le fonctionnement de tout AG peut être décrit par le principe illustré par la figure 2.4.

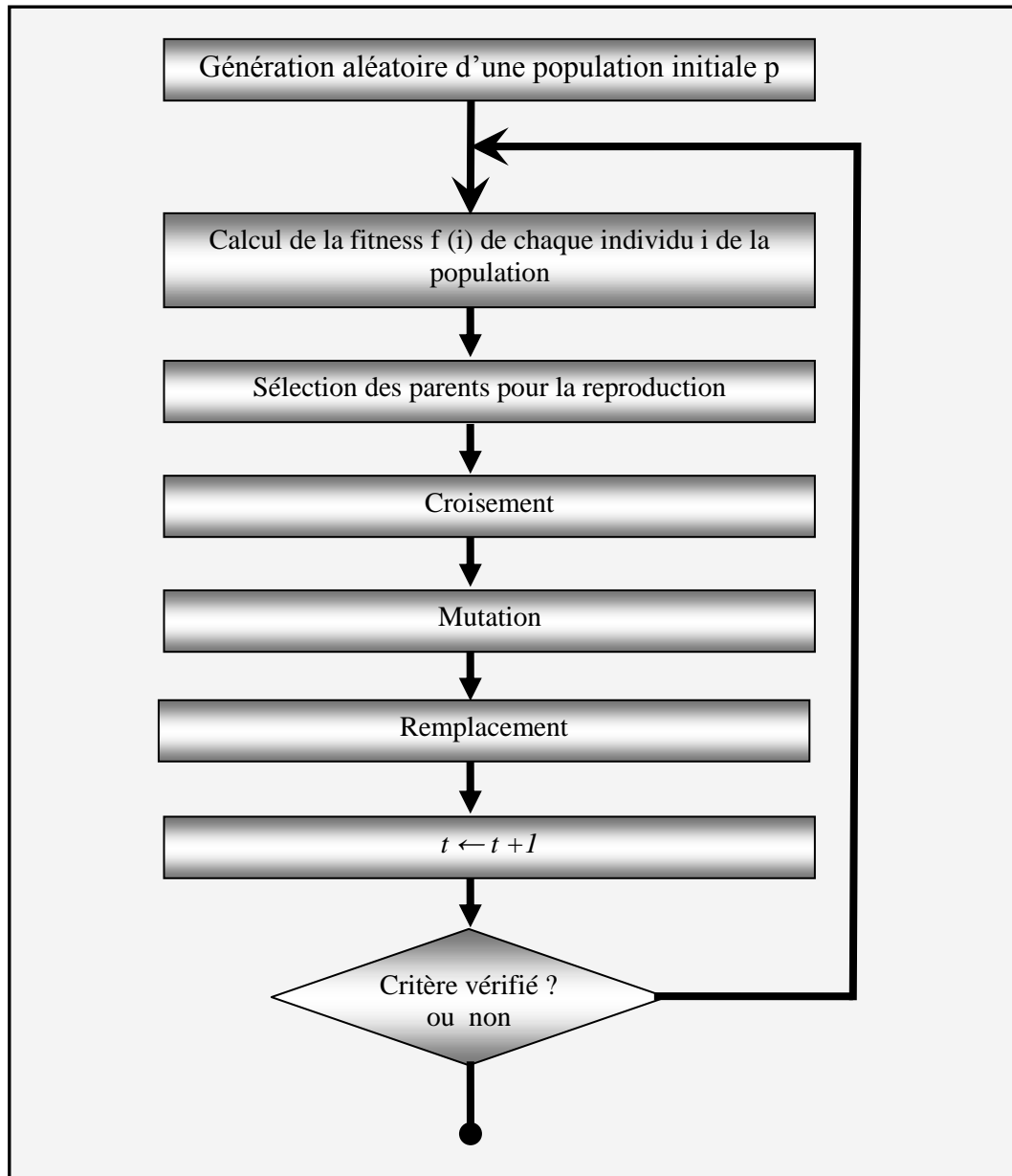


Figure 2.4. Principe de fonctionnement d'un AG

En règle générale, pour pouvoir exploiter efficacement le potentiel d'un AG pour la résolution d'un problème donné, il est demandé de prendre en compte les points suivants :

Comment représenter les individus du problème en question ? Quelle fonction d'évaluation doit être utilisée pour la sélection ? Comment est le croisement effectué ainsi que les mutations ? Quels sont les valeurs des paramètres entrant en jeu (taille de la population, probabilités des opérateurs) ? Quand est ce que l'algorithme doit s'arrêter (critère d'arrêt) ?

Avant d'expliquer chacune des étapes figurant dans le schéma ci-dessus, nous allons d'abord préciser quelques choix techniques offerts pour coder un chromosome.

4.1. Codage des chromosomes

Les individus de la population doivent être codés selon une représentation spécifique. Le choix d'adopter un tel codage ou un autre est une question qui dépend des caractéristiques du problème présent. Chaque paramètre d'une solution du problème traité est assimilé à un gène. Parmi les techniques les plus fréquemment utilisées pour coder les individus, on distingue :

- **Codage binaire** : Proposé par Holland lui-même [6] dans sa version originale des AG où les individus ont été codés sous forme de chaînes binaires, chaque gène est représenté par une sous chaîne binaire constituée d'un nombre m de bits fixé en fonction de la précision souhaitée. Certainement, ce codage est le plus répandu en raison de ses nombreux avantages notamment la manipulation des gènes.

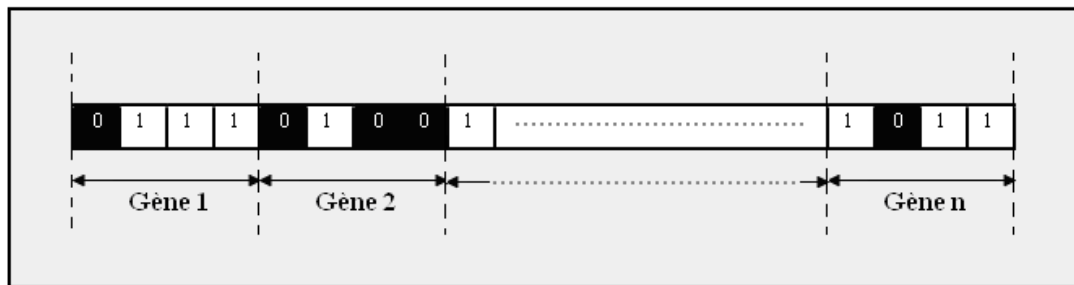


Figure 2.5. Structure d'un chromosome en codage binaire

- **Codage en nombres réels** : Le codage binaire présente un inconvénient majeur : que se passe-t-il si le nombre de paramètres d'une solution est assez grand ? Sans aucun doute il sera très fastidieux et très difficile de gérer les chromosomes. Une solution possible est l'utilisation des nombres réels pour coder les gènes des individus de la population, ce codage est mieux adapté aux problèmes imposant une précision. Techniquement cela peut être réalisé par un vecteur dont les coordonnées sont les paramètres du problème à résoudre.

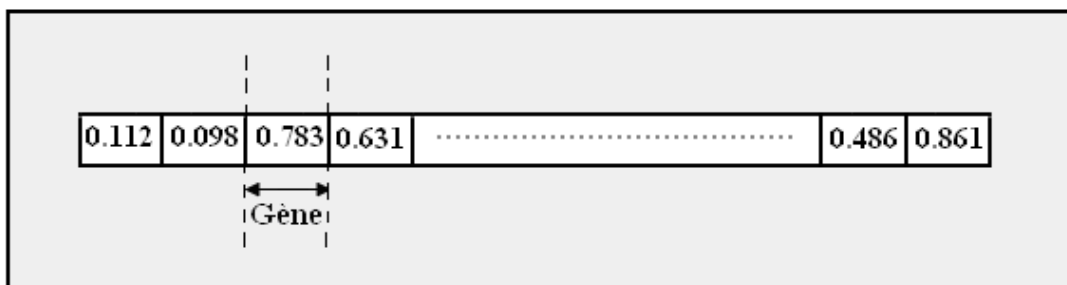


Figure 2.6. Structure d'un chromosome en codage en nombres réels

- **Codage à base N** : Un autre inconvénient majeur du codage binaire est présent : que se passe-t-il si les gènes correspondent à un ensemble discret de valeurs dont le cardinal n'est pas une puissance de 2 ? Il sera possible que des combinaisons binaires ne correspondant pas à un codage valide puissent apparaître. Pour surmonter ce problème, d'autres chercheurs ont inventé un nouveau codage appelé codage à base n où un chromosome est essentiellement constitué d'un ensemble d'éléments qui sont des chiffres exprimés dans une base de numération n en permettant de représenter n valeurs discrètes.

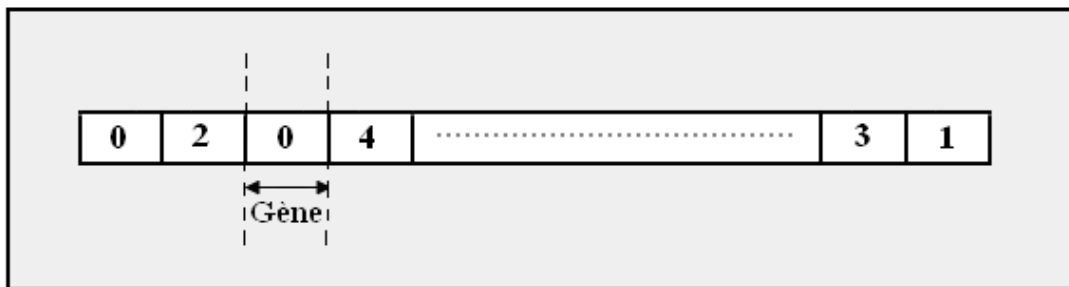


Figure 2.7. Structure d'un chromosome en codage à base n ($n = 5$)

4.2. Initialisation de la population

C'est une phase préparatoire pour générer la population initiale à partir de laquelle démarre l'AG son exécution. On commence d'abord par fixer l'espace de recherche ou encore le domaine de définition des individus. Ensuite, des fonctions pour le codage et le décodage des individus doivent être établies. Elles servent à associer le génotype d'un individu qui est inclus dans l'espace de recherche à son phénotype qui est la valeur de la fonction pour cet individu. Une fois la structure des individus est déterminée, il ne reste qu'à générer les individus de la population initiale. Elle doit également contenir des individus qui soient bien répartis dans l'espace des solutions ce qui permet une variété dans le matériel génétique. La façon la plus simple et la plus naturelle est d'initialiser aléatoirement les valeurs des gènes des individus initiaux, le plus souvent, suivant une distribution uniforme. Le choix du codage et de la taille initiale de la population doit être rigoureusement défini étant donné leur influence sur l'exécution ultérieurement.

4.3. Evaluation des individus

Cette phase a pour objectif de quantifier la qualité de chaque chromosome dans la population pour la reproduction. L'évaluation se fait selon une fonction dite d'adaptation qui associe à

chaque phénotype une valeur d'adaptation, elle permet donc de noter les individus de la population. Plus cette note est élevée, plus cet individu est plus adapté à survivre permettant ainsi à ses descendants d'hériter son matériel génétique. Le choix de la fonction d'évaluation est d'une grande importance : d'une part les performances de l'algorithme dépendent essentiellement d'elle du fait qu'elle est exécutée à chaque génération sur tous les individus, et d'autre part elle doit être capable d'évaluer les individus de manière efficace et sélective.

4.4. Sélection

Une fois réalisée l'évaluation de la génération, on opère une sélection à partir des valeurs d'adaptation. Seuls les individus passant l'épreuve de sélection peuvent se reproduire. Notons que les étapes de sélection et de remplacement sont indépendantes de l'espace de recherche. On distingue deux types de sélection :

- **La sélection déterministe** : Seuls les meilleurs individus seront toujours sélectionnés, les autres sont totalement écartés.
- **La sélection stochastique** : Les meilleurs individus sont toujours favorisés mais de manière stochastique ce qui laisse une chance aux individus moins adaptés pour participer à former la prochaine génération.

Pratiquement, il y a plusieurs techniques de sélection parmi lesquelles on cite :

- **Sélection par élitisme** : Les individus de la population sont triés selon leurs valeurs d'adaptation. Seuls les premiers individus de la moitié supérieure de la population seront sélectionnés pour se reproduire. La variance de cette technique est nulle mais en revanche la pression de sélection est très élevée. En effet, les chromosomes faibles n'ont en principe aucune chance pour être sélectionnés contrairement à ceux forts qui sont toujours sélectionnés.
- **Sélection par tirage de roulette** : Imaginons une roulette divisée en cases, similaire à une roulette de fortune, chaque case correspond à un individu et dont la taille dépend de son adaptation à son environnement. On lance la roulette puis on sélectionne l'individu possédant la case où elle est tombée. Le processus sera répété autant de fois qu'il y a d'individus à sélectionner. Il est clair que cette technique présente une forte

variance : avec un peu de malchance, les individus très mauvais peuvent être choisis autant de fois qu'il y a de places pour passer à la génération suivante et de même, il se peut qu'aucun des bons individus ne soit sélectionné.

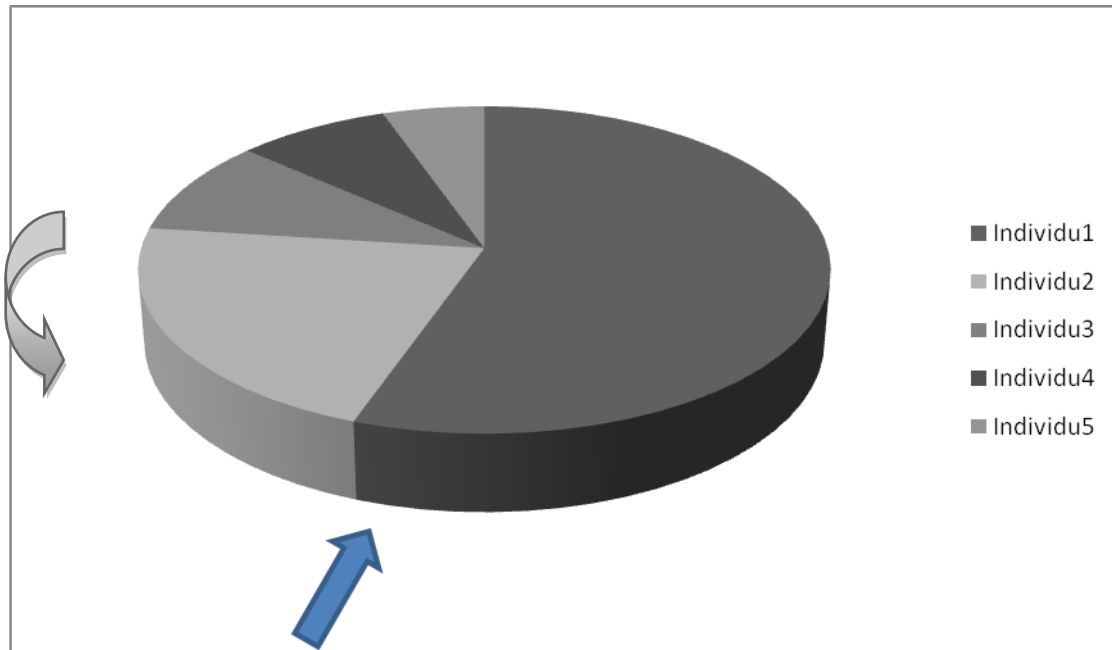


Figure 2.8. Tirage par roulette.

- **Sélection par tournois :** Deux individus sont choisis au hasard pour se combattre, un et un seul entre eux soit le gagnant (en comparant leurs valeurs d'adaptation) pour pouvoir participer à la reproduction. On répète le processus autant de fois qu'il y a d'individus à sélectionner. On peut également généraliser cette technique pour n individus. La variance de cette technique est élevée. En effet, on peut accorder plus ou moins de chance aux individus peu adaptés. Si par exemple le nombre de participants est assez grand, un individu faible sera, presque toujours, sûrement écarté. Pour combler cette insuffisance, il est possible de compter sur un mécanisme stochastique en introduisant un facteur de probabilité le plus souvent compris entre 0.5 et 1 selon lequel les individus les plus adaptés seront acceptés, ceci permet de contrôler la pression de sélection si nécessaire.
- **Sélection uniforme :** La sélection se fait sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité de $1/P$ pour être sélectionné, P étant le nombre d'individus de la population. La variance de cette technique est très forte : avec un peu

de malchance, il sera possible que l'ensemble de la population ne contienne que des individus très mauvais.

4.5. Croisement

Après avoir sélectionné un certain nombre d'individus, ils seront inclus dans une nouvelle génération intermédiaire puis seront aléatoirement répartis en couples. A ce moment là, chaque couple peut commencer à se reproduire, permettant par-là la construction d'une nouvelle génération. Les parents doivent passer leurs caractéristiques génétiques à leurs descendants en copiant et recombinant leurs matériels génétiques de façon à ce que chaque couple donne lieu à la naissance de deux descendants. L'opérateur de croisement assure le brassage du matériel génétique et l'accumulation des mutations favorables ce qui permet de créer de nouvelles combinaisons des paramètres des solutions. Il existe plusieurs façons pour réaliser un croisement entre deux chromosomes :

- **Croisement en un point** : On choisit au hasard un point pour couper les deux chromosomes de chaque couple puis on échange les fragments situés après le point de coupure permettant la création de deux nouveaux génotypes. Notons que le croisement peut également ne pas s'effectuer au niveau des gènes ce qui rend possible à un chromosome d'être coupé au milieu d'un gène.

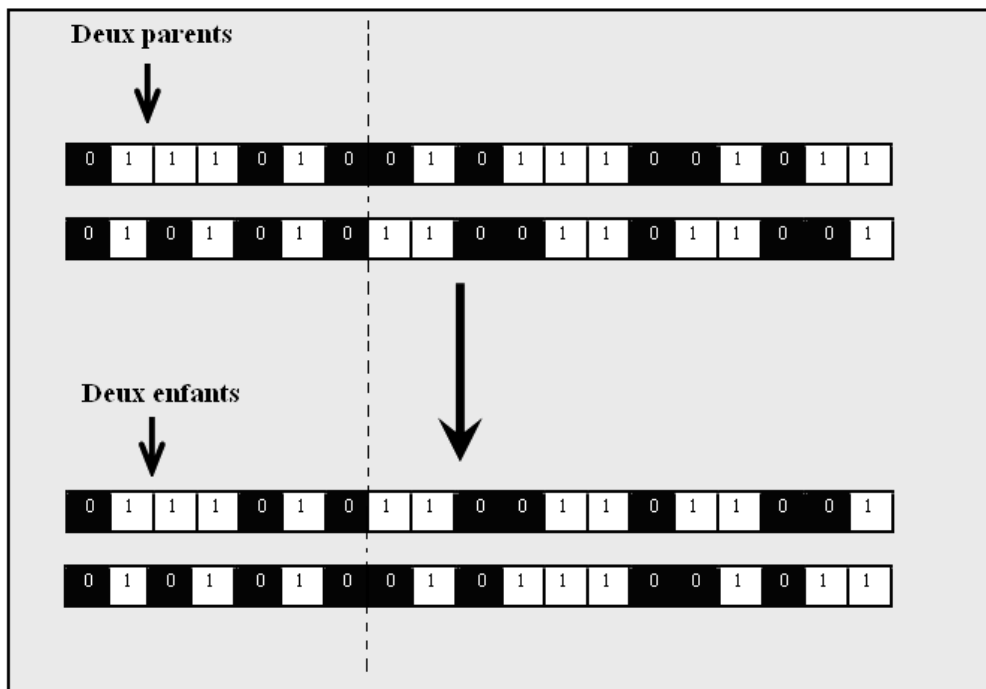


Figure 2.9. Croisement en un point

- **Croisement en deux points** : On choisit au hasard deux points de croisement puis on échange les fragments situés entre ces deux points. Cette définition peut également se généraliser pour effectuer un croisement à n points.

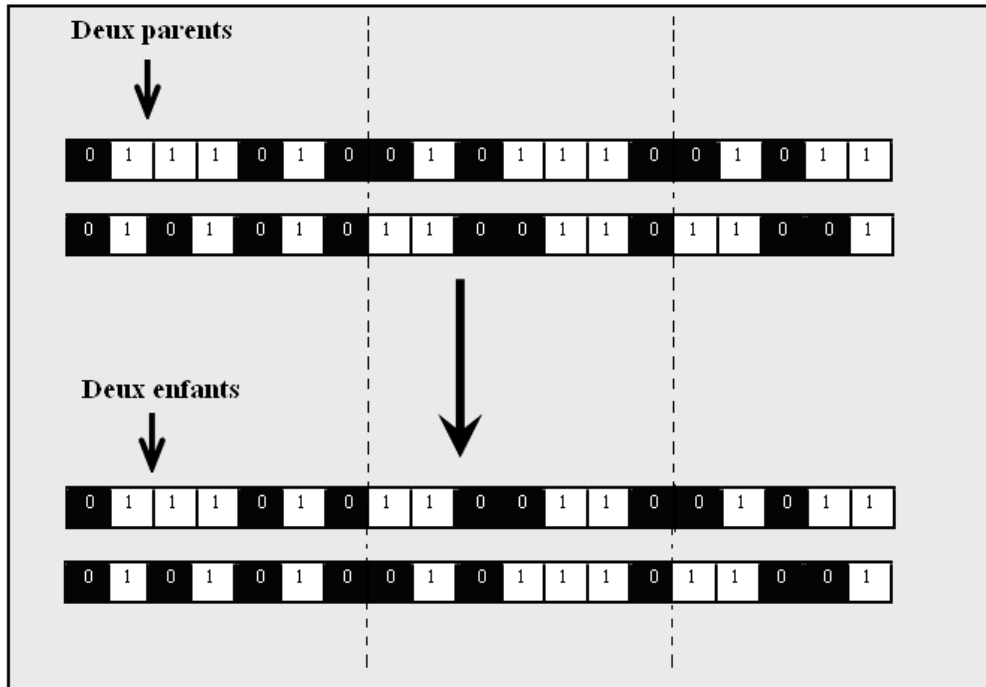


Figure 2.10. Croisement en deux points

4.6. Mutation

Nous définissons une *mutation* comme étant la modification aléatoire, qui n'est pas issue d'une opération de croisement, de la valeur d'un paramètre du dispositif d'un chromosome avec une faible probabilité, typiquement entre 0.01 et 0.001 .

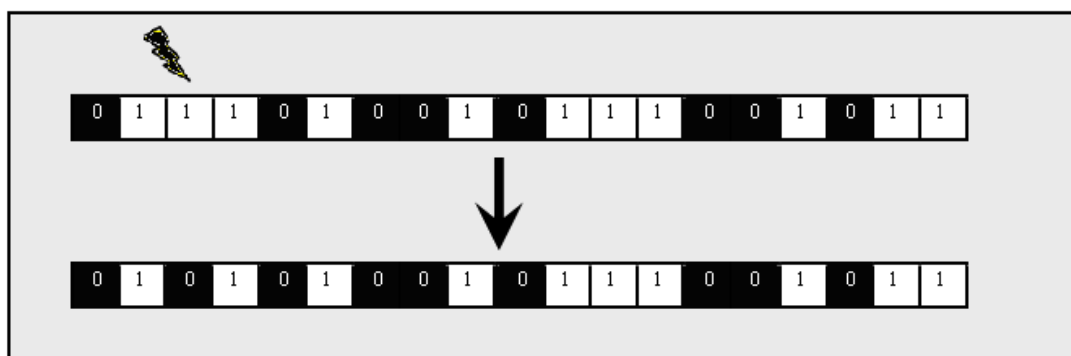


Figure 2.11. Opérateur de mutation

Pourquoi une mutation ? La raison est d'éviter le phénomène de *dérive génétique*. Certains gènes favorisés par des erreurs stochastiques peuvent se répandre au détriment des autres et seront ainsi présents au même endroit sur tous les génotypes. Ceci implique que la convergence de l'algorithme aura lieu vers un optimum local et on parle alors de *la convergence prématurée*. Les mutations permettent alors un maintien en introduisant constamment de nouveaux gènes conduisant à une *diversité génétique*. Un autre point essentiel que ne doit être négligé, supposons par exemple que l'évolution, après un certain nombre d'itérations, a conduit à une population dont la plupart des individus ont convergé vers l'optimum global, à ce moment là, inutilement de faire seulement croiser les chromosomes car ils sont souvent quasi identiques. Les mutations conduisent alors à rapprocher les individus vers l'optimum global d'autant que le permet la précision du codage.

Le choix de la probabilité de mutation est très important car si elle sera trop élevée, on risque de ne pas voir une bonne convergence, et si elle sera trop faible, ça réduit d'autant son effet. Une conclusion est que le croisement est essentiellement utile pour explorer globalement et entièrement l'espace de recherche tandis que les mutations sont principalement utiles pour la recherche locale et l'optimisation des solutions déjà rapprochées.

4.7. Remplacement

A ce stade, une nouvelle génération doit naître à partir des parents croisés et mutés de la génération précédente qui ont formé les nouveaux enfants, il en résulte une amélioration de la population. Lors de sa construction, deux cas peuvent se figurer : les nouvelles solutions peuvent remplacer totalement ou partiellement les anciennes solutions.

- **Remplacement stationnaire** : Les enfants remplacent automatiquement les parents sans tenir compte de leurs performances respectives.
- **Remplacement élitiste** : On garde au moins l'individu possédant les meilleures performances d'une génération à la suivante.

Une fois la nouvelle génération créée, le processus d'évolution doit commencer à nouveau dans le but de construire une autre génération dont les individus seront de plus en plus adaptés. L'algorithme s'arrête quand le meilleur individu d'une génération a dépassé un certain

seuil ou quand un certain nombre d'individus sont proches d'une solution locale optimale (convergence partielle).

En conclusion, nous pouvons dire que la prolifération de la popularité des AG est essentiellement issue de leur généricité qui permet la résolution de plusieurs classes de problèmes. Pour preuve, il suffit de voir le grand nombre d'articles consacrés à la résolution de divers problèmes notamment les problèmes combinatoires et ceux d'apprentissage. Une autre caractéristique fondamentale que possèdent les AG, celle du parallélisme intuitif où l'on peut traiter plusieurs chromosomes en même temps permettant par-là la construction des systèmes hautement performants. L'intérêt de leur utilisation peut donc être résumé par la formule : "*puissance + généricité*".

Partie III : Vers des automates cellulaires évolutifs

Jusqu'ici nous avons abordé deux concepts assez importants d'un point de vue technique et formel. Le premier concerne le modèle abstrait d'AC, le second concerne la discipline d'AG. On peut se demander s'il existe une relation entre ces deux concepts. La réponse à cette question se traduit principalement par l'émergence d'un nouveau concept relativement jeune appelé 'automate cellulaire évolutif', un concept qui semble être hybride et donc touchant à ces deux premiers. La partie qui suit a pour but de l'étudier en commençant par définir son cadre générale puis présenter ses aspects techniques et formels tout en levant l'ambiguïté par l'étude d'un exemple applicatif.

1. Automates cellulaires évolutifs

D'une manière informelle, un ACEV n'est rien d'autre qu'un AE manipulant une population dont les individus représentent des AC. Vu leur popularité incroyable par rapport aux autres AE, les AG ont pour autant été utilisés dans la littérature pour évoluer des populations d'AC. Pour cette raison nous avons décidé de n'aborder, par la suite, en détail que *l'évolution des AC par des AG*.

Nous avons vu dans la partie précédente que l'un des points clés des AG est comment représenter les individus de la population, en particulier comment faire pour que la structure d'un AC soit capable d'entrer dans le moule des chromosomes, le reste sera en principe facile car les autres opérations qu'effectue un AG dépendent principalement de la structure du chromosome indépendamment de ce qu'il représente. La partie d'un AC que doit être représentée par un chromosome est sa fonction de transition. Nous allons voir comment cela est techniquement possible.

2. D'un automate cellulaire vers un chromosome

Soit A un AC, f est sa fonction de transition et V est l'ensemble des états cellulaires. On note que f est une fonction de V^n dans V , n étant le nombre de cellules par voisinage. Il en résulte que l'ensemble des fonctions de transitions s'il est défini de manière extensionnelle est fini (on suppose ici que l'ensemble V est fini). Le cardinal de l'ensemble V^n est égal à $\text{card}(V)^n$ ce qui représente en fait le nombre de toutes les configurations possibles du voisinage.

Formellement, le cardinal de l'ensemble des fonctions de transitions est donc donné par la formule 2.2.

$$(f: V^n \rightarrow V) \wedge (\text{card}(V^n) = \text{card}(V)^n) \rightarrow \text{card}(\{f\}) = \text{card}(V)^{\text{card}(V)^n}$$

Formule 2.2. Cardinal de l'ensemble des fonctions de transitions

Par exemple, si l'on considère un automate unidimensionnel à deux états et un voisinage local de 7 cellules, nous avons donc $\text{card}(V) = 2$ et $n = 7$, d'où :

$$\text{card}(\{f\}) = 2^{\text{card}(V)^n} = 2^{2^7} = 2^{128}$$

La fonction de transition peut également être représentée par une table appelée « *lookup table* » qui met en correspondance toutes les configurations possibles du voisinage avec le nouvel état associé [16]. La taille de cette table est le cardinal de l'ensemble V^n . A chaque configuration possible du voisinage est associée une position dans la table de correspondance permettant par-là l'indexation des états de sortie de la fonction de transition. Cette structure permet de passer facilement vers une structure d'un chromosome où chaque case de la table, qui représente un état de sortie d'une configuration du voisinage, prenant la valeur vi et dont la position est de i correspond à un gène du chromosome dont le locus est de i et l'allèle prenant la valeur vi . Les gènes peuvent également être codés, selon le besoin, par un codage binaire, à base n ou encore un autre codage. En fin de compte, on peut dire que chaque gène est indexé par un locus correspondant à une configuration possible du voisinage.

3. Construction d'un automate cellulaire évolutif

3.1. Initialisation de la population

On commence d'abord par fixer l'espace de recherche, cet espace correspond à l'ensemble de toutes les fonctions de transitions possibles. Viennent ensuite les fonctions de codage et de décodage. Les premières servent à coder une fonction de transition sous forme d'un chromosome. Le choix du codage dépend essentiellement de l'ensemble des états cellulaires V , par exemple pour un automate à deux états le codage binaire semble être le plus approprié tandis que pour un automate ayant plus de deux états le codage à base n semble être le plus approprié. Les deuxièmes servent à interpréter les chromosomes afin de permettre leur évaluation. Comme tout AG, la population initiale des chromosomes doit être créée de façon à ce que les individus la constituant soient bien répartis par rapport à l'espace de recherche, conduisant à une diversité génétique. A chaque individu de la population est associé un

ensemble de configurations initiales qui assure le déroulement de l'automate représenté, elles peuvent également être générées aléatoirement ou arbitrairement (configurations initiales particulières).

3.2. Evaluation des individus

L'évaluation de chaque chromosome se fait par l'évolution de son ensemble des configurations initiales pendant un certain nombre d'itérations. L'ensemble des configurations finales sera évalué pour pouvoir déterminer la valeur d'adaptation de cet individu.

3.3. Opérations génétiques

Un point fort des AG est que les opérations génétiques opèrent sur les chromosomes sans avoir recours à ce qu'ils représentent. Les opérations génétiques se déroulent normalement comme elles sont décrites dans la partie précédente. Le seul problème qui reste est la détermination de comment ces opérations seront effectuées. Par exemple, comment sélectionne-t-on les chromosomes les plus adaptés ? Utilise-t-on un croisement en un point, en deux points ou en plusieurs points ? Quelle probabilité utilise-t-on pour les mutations ? Quelle stratégie de remplacement choisit-on : un remplacement stationnaire ou un remplacement élitiste ?

4. Etude de cas

Nous allons présenter dans cette section un exemple applicatif pour la mise en œuvre d'un ACEV. On considère ainsi un AC bidimensionnel à deux états dont le voisinage est de 9 cellules constitué d'une cellule au centre et ses huit voisines immédiates.

Nous avons donc $\text{card}(V) = 2$ et $n = 9$, d'où :

$$\text{card}(V^n) = (2^9) = 512 \text{ et } \text{card}(\{f\}) = 2^9 = 512$$

f et V étant la fonction de transition et l'ensemble des états cellulaires respectivement.

Le but étant de résoudre le problème de la classification de densité $pc = 1/2$. Pour cette tâche l'automate décide pour chaque configuration initiale de cellules noires ou blanches, s'il y a plus de cellules noires ou plus de cellules blanches. Quand il y a plus de cellules noires, l'automate doit tomber dans une configuration où toutes les cellules sont noires. On veut donc utiliser un AG pour essayer obtenir par évolution des automates capables de réaliser cette tâche. L'AG doit également s'arrêter lorsque le meilleur individu est capable de réaliser la tâche demandée ou bien l'algorithme a itéré pendant un certain nombre de générations.

4.1. Codage des chromosomes

A Chaque fonction de transition on associe une table de correspondance dont la taille est de 512 positions, chacune correspond à une configuration bien déterminée du voisinage. La figure 2.11 représente une façon possible pour faire correspondre une configuration du voisinage à sa position dans la table de correspondance et par conséquent d'en déduire le locus du gène correspondant dans le chromosome.

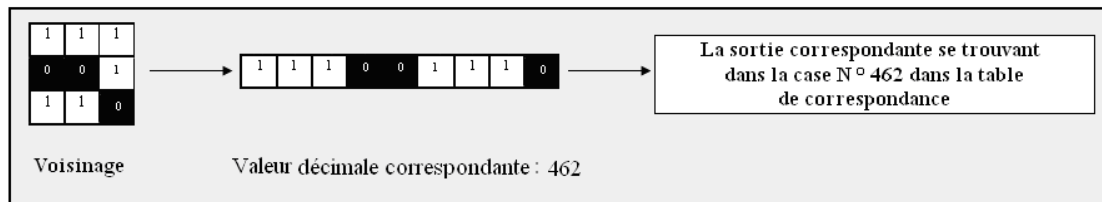


Figure 2.12. Calcul des locus des gènes

Le chromosome résultant est donc composé de 512 gènes au total où chaque gène dont le locus est de i correspond à la sortie se trouvant dans la position i de la table de correspondance. La figure 2.12 illustre la structure d'un chromosome.

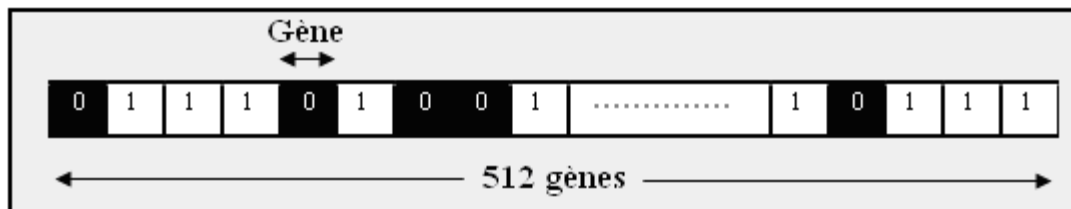


Figure 2.13 Structure d'un chromosome [512 bits]

4.2. Initialisation de la population

Nous avons vu qu'il existe 2^{512} fonctions de transitions possibles, cet ensemble correspond à l'espace de recherche manipulé. Chaque individu est codé sur un chromosome qui est une chaîne binaire de 512 bits. La fonction de décodage sert à permettre l'évolution de chaque automate représenté par un individu pour générer ses configurations finales. Les individus sont aléatoirement générés cependant, ils sont contraints d'être uniformément répartis sur λ , λ étant le paramètre de Langton vu dans le chapitre précédent. A chaque individu on associe un ensemble de configurations initiales générées aléatoirement mais qui sont forcées d'être uniformément réparties sur $\rho \in [0.0, 1.0]$ de façon à ce que pour la moitié $\rho < \rho_c$ et $\rho > \rho_c$ pour l'autre moitié.

4.3. Evaluation des individus

L'évaluation de chaque chromosome se fait par son évolution sur l'ensemble de ses configurations initiales pendant M itérations. On attribue comme note à un individu la fraction de ses configurations initiales pour lesquelles la réponse est correcte.

4.4. Opérations génétiques

- **Sélection** : On adopte une sélection déterministe suivant la stratégie d'élitisme, les individus sont donc triés selon leurs valeurs d'adaptation où seuls les meilleurs individus seront sélectionnés pour la reproduction.
- **Croisement** : On adopte un croisement en un point. Une position aléatoire est donc choisie pour réaliser l'opération. On remarque ici que le croisement ne s'effectue qu'au niveau des gènes c'est-à-dire qu'il est impossible qu'un chromosome soit coupé au milieu d'un gène du fait qu'il est équivalent à un bit.
- **Mutation** : La probabilité pour qu'un gène soit muté est choisie d'être de 0.01 , elle sert à inverser la valeur des bits du chromosome selon cette probabilité.
- **Remplacement** : Nous avons choisi le remplacement stationnaire, les nouveaux chromosomes remplacent donc directement ceux écartés sans tenir compte de leurs valeurs d'adaptation.

Conclusion

En conclusion, les AC sont donc des systèmes dynamiques définis à base d'un modèle mathématique rigoureux : des règles extrêmement simples pouvaient émerger un comportement global résultant très complexe. Leur puissance se reflète par la capacité d'exhiber tous les types de comportements dynamiques (périodique, point fixe, chaotique et complexes) et par la capacité de réaliser toute sorte de calcul (universalité). C'est ce point qui semble être le secret de l'intérêt croissant par la communauté des chercheurs de la physique, des sciences de la complexité et de l'informatique.

La difficulté de concevoir un AC pour présenter un comportement spécifique ou effectuer une tâche particulière a malheureusement limité leurs applications du fait que les règles de transitions devaient être soigneusement produites à la main. Pour preuve, il suffit de se souvenir que Conway avait consacré deux ans de travail continu avant de parvenir à trouver les règles de *Life*, ce n'était pas dû à un manque de connaissances concernant les AC mais plutôt c'était dû à la difficulté de trouver des règles simples conduisant à des comportements imprévisibles. Pour surmonter ce problème, il était nécessaire de compter sur un moyen plus efficace : automatiser le processus de conception des AC, une tâche qui est très semblable à résoudre un problème combinatoire.

L'approche des AG a connu depuis quelques décennies un intérêt croissant notamment pour la résolution des problèmes combinatoires nécessitant une exploration d'un espace de recherche. Ils sont, également, par nature, hautement parallèles dans la mesure où ils peuvent simuler l'évolution de tout un ensemble de solutions codées sous forme de chromosomes.

Le couplage de ces deux techniques a donné lieu à une nouvelle approche pouvant résoudre le problème posé : les ACEV qui consistent à évoluer des populations d'AC par des AE. Pour le faire avec un AG, il suffit de pouvoir représenter un AC par un chromosome, nous avons vu que cela est relativement facile à réaliser grâce à un système de conversion permettant par-là de tirer profit de la puissance de recherche de tel algorithme pour trouver l'AC souhaité.

*Vers des
automates
cellulaires
évolutionnaires
quantiques*

Chapitre
3

11. Introduction
12. Les ordinateurs quantiques entre la réalité et l'espoir
13. Introduction à l'informatique quantique
14. Algorithmes génétiques quantiques (AGQ)
15. Extension pour évoluer des AC
16. Performances des AGQ
17. Algorithme proposé basé sur les AGQ pour l'évolution des AC
18. Conclusion

Vers des automates cellulaires évolutionnaires quantiques

1. Introduction

La partie précédente était consacrée à l'étude du concept d'ACEV d'un point de vue technique et formel. Nous avons abordé le sujet en étudiant l'évolution des AC par des AG. Dans cette partie nous allons proposer notre contribution apportée sur ce concept en l'étendant pour supporter *l'évolution des AC par des algorithmes génétiques quantiques (AGQ)*. Deux raisons sont derrière ce choix : d'une part nous avons remarqué un intérêt croissant pour l'étude des AGQ afin de résoudre des problèmes d'optimisation et d'autre part **l'absence totale des travaux traitant l'évolution des AC par des AGQ**. Ceci nous a encouragés à prendre cette voie permettant ainsi d'avoir une originalité pour notre travail de recherche.

L'une des caractéristiques fondamentales que possède un AG est celle du parallélisme intuitif avec lequel on peut traiter plusieurs chromosomes en même temps. La question qui se pose ici est donc : jusqu'à quelle mesure les AGQ peuvent ils s'avérer convenables à des traitements parallèles ?

Nous avons choisi d'aborder la question en présentant d'abord l'intérêt pour l'informatique quantique ainsi qu'une initiation au domaine. Ensuite, nous allons étudier l'existence du parallélisme, c'est-à-dire différencier formellement les tâches pouvant s'exécuter en parallèle de celles s'exécutant seulement en séquentiel afin d'extraire le potentiel du parallélisme exploitable par cette approche. Enfin, pour aller encore plus vite, nous proposons d'adopter une solution pouvant aider à augmenter les performances globales du processus évolutionnaire afin de mieux explorer des espaces de recherche d'AC comme nous le verrons plus tard.

2. Les ordinateurs quantiques entre la réalité et l'espoir

« Cofondateur de la société Intel, Gordon Moore avait affirmé dès 1965 que le nombre de transistors par circuit de même taille allait doubler, à prix constants, tous les ans. Il rectifia par la suite en portant à dix-huit mois le rythme de doublement. Il en déduisit que la puissance des ordinateurs allait croître de manière exponentielle, et ce pour des années. Il avait raison. Sa loi, fondée sur un constat empirique, a été vérifiée jusqu'à aujourd'hui. Il a cependant déclaré en 1997 que cette croissance des performances des puces se heurterait aux environ de 2017 à une limite physique : celle de la taille des atomes. D'ici là, nos ordinateurs seront environ 1500 fois plus puissants qu'aujourd'hui ! » [44]

Que ce soit à court ou à long terme, nous allons tomber sur une situation inévitable : les composants électroniques deviendront de la taille des atomes ce qui conduit à aboutir à un stade final que les développements en performance ne peuvent dépasser. Une solution à ce problème, paraissant viable, semble être le changement du paradigme pour se tourner vers l'informatique quantique. Pour preuve, la société D-Wave a déjà fait la présentation officielle d'Orion en 2007, le premier processeur au monde composé de 16 qubits et censé pouvoir être commercialisé.

« Le fonctionnement d'Orion fait appel aux principes de la mécanique quantique. D-Wave a affirmé qu'Orion est capable d'exécuter 64000 calculs simultanément. D'ailleurs pour démontrer la puissance de son produit, les représentants ont mis en avant un problème non déterministe polynomial, dit du voyageur de commerce. Pour ce problème, il est demandé de calculer l'itinéraire d'un vendeur partant depuis une ville et arrivant à la même ville en passant une fois par différentes villes, le but étant de trouver l'itinéraire le moins cher. D-Wave a pris l'exemple de la Suède et de ses 24978 villes qui demanderaient 85 ans de calculs pour un ordinateur classique. Orion ne demande que quelques cycles. Orion tourne néanmoins à une fréquence bien inférieure de celle des processeurs classiques d'aujourd'hui et la firme parle d'un processeur de 1000 qubits plus rapide pour la fin 2008. Néanmoins, le but de cette démonstration n'était pas d'exhiber des performances, mais de montrer que les calculateurs quantiques deviennent petit à petit une réalité et qu'ils vont continuer sur le chemin du progrès » [56].

Alors, faut-il déjà considérer la mise à l'index des ordinateurs traditionnels ? Selon Christophe Jacquemin la réponse à cette question n'est si évidente pour l'affirmer :

« A notre avis, cela semble être bien prématuré. D'ailleurs l'entreprise elle-même déclare que le calcul quantique ne doit pas être vu ici comme un remplacement des calculateurs digitaux, mais comme l'apport de nouvelles possibilités, par exemple dans le cas de certains types de calculs liés à la finance, la biologie, la chimie, la sécurité (biométrie notamment), la défense, la logistique...

Le prototype présenté est aujourd'hui 100 fois plus lent qu'un calculateur numérique courant. Pour chaque problème considéré, Orion fait tourner le calcul de multiples fois (à cause du bruit de fond qui fait que la puce peut retourner quelquefois des résultats erronés) et détermine quelle réponse a la probabilité la plus élevée d'être exacte. La meilleure solution est déterminée par le niveau d'énergie le plus bas du système. La deuxième meilleure solution - en raison de la nature physique du calcul - est le deuxième niveau bas d'énergie, etc. La solution qui minimise l'énergie et qui ressort le maximum de fois est considérée comme étant la bonne (un test sur des problèmes où la réponse correcte était déjà connue a montré que la majorité des réponses données par l'ordinateur quantique était toujours exactes). Deux objectifs sont alors poursuivis par D-Wave :

- Parvenir à une solution plus précise, avec la même rapidité qu'un calculateur numérique.*
- Obtenir une solution de la même exactitude mais de façon plus rapide qu'un calculateur numérique. » [55].*

A notre tour, nous pouvons facilement constater que le paradigme d'informatique quantique se heurte à de nombreuses difficultés et problèmes qui restent encore à résoudre. Comment par exemple augmenter le nombre de qubits sachant que la décohérence rend les systèmes trop fragiles pour être exploitables, la correction des erreurs semblant être un problème prégnant ? Comment programmer ce type d'ordinateurs pour pouvoir produire un résultat déterministe à partir de hasard ? D'autres questions peuvent également être posées sans trouver de réponses. Est-ce la fin des temps pour le développement informatique ?

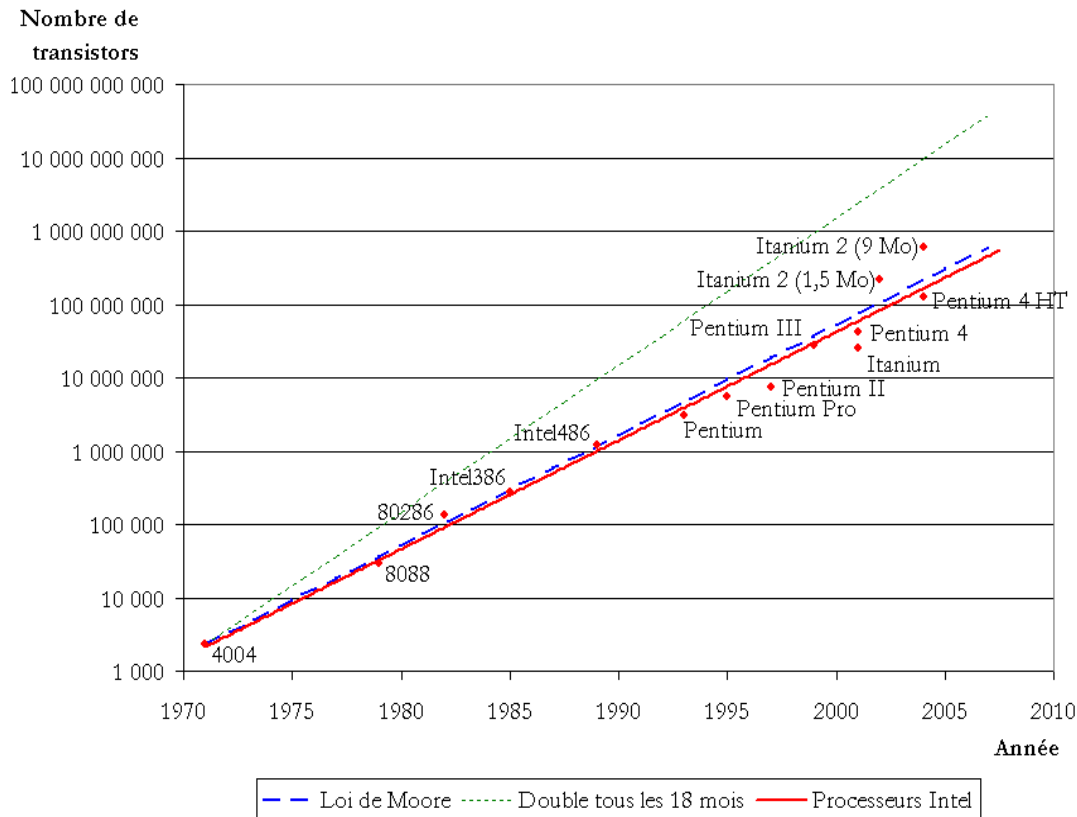


Figure 3.1. Développement des circuits selon la loi de Moore [43]

La réponse à cette question est loin d'être affirmative. En effet, malgré tous ces obstacles, les recherches effectuées sur ce sujet nous font sentir qu'elles ne visent qu'à confirmer la progression croissante du paradigme naissant. A notre avis, il semble que Christophe Jacquemin avait raison lorsqu'il a affirmé qu'il est encore prématuré de confirmer la mort totale de l'intérêt des ordinateurs classiques. Mais par ailleurs, il semble aussi que les objectifs poursuivis par D-Wave ne sont que des objectifs transitoires touchant à l'aspect performance en attendant la résolution des problèmes majeurs qui entravent la construction fiable des ordinateurs quantiques, permettant par là d'ouvrir la voie à une nouvelle génération d'ordinateurs. En fait, on doit garder à l'esprit que Moore a confirmé que la croissance des performances des puces se heurterait aux environs de 2017 à une limite physique, une conclusion selon laquelle la plupart des constructeurs seront disparus si la situation demeure telle qu'elle est actuellement. La tendance informatique doit donc s'orienter vers de nouveaux paradigmes pour assurer la continuité du développement.

3. Introduction à l'informatique quantique

3.1. Source d'inspiration

« Lorsque l'on parle d'un ordinateur quantique on emmène rapidement vers des concepts de la physique quantique. Le premier point important en physique quantique est que les propriétés des systèmes physiques, électrons par exemple, sont quantifiées. Le comportement de l'énergie est modélisé à une très petite échelle à l'aide des quanta, quantités discontinues. Une propriété quantifiée souvent utilisée dans la recherche sur l'informatique quantique est le spin qui ne peut prendre, pour un électron, que les valeurs $+1/2$ ou $-1/2$ ou encore (+, -) en abrégé, et on pourrait alors parler aussi de 0 et de 1 comme en informatique classique.

Un deuxième point est la superposition d'états. Le chat quantique de Schrödinger peut être à la fois vivant et mort, le spin peut être à la fois (+) et (-). Seule la mesure détermine le signe (+) ou (-) avec une certaine probabilité pour chacun, cela détruit la superposition permettant ainsi de lever l'ambiguïté. Par analogie avec le bit d'information en informatique classique (1 ou 0), cet état intriqué (+ / -) est appelé qubit. C'est ce point qui semble être le secret d'intérêt croissant par l'informatique quantique : l'information n'est plus limitée à 2 valeurs, mais peut s'établir dans un continuum.

Un troisième point paraissant aussi important est l'intrication. Un couple d'électrons en physique quantique, équivalent d'un couple de qubits en informatique quantique, conduit à avoir 4 états : (++, --, +-, -+). Par superposition, il est alors possible de créer une paire dans un état dit intriqué, chacun avec une probabilité associée. Pour un ordinateur quantique, cette notion est très importante car ces états jouent le rôle de flux d'information capables de subir en parallèle des traitements » [47].

3.2. Notions de base

3.2.1 Les qubits

Les ordinateurs classiques fonctionnent autour d'une architecture binaire où les unités d'information, les bits, peuvent prendre deux valeurs, 0 ou 1 signifiant que le courant passe ou pas. En informatique quantique, cette notion est carrément écartée où les bits sont remplacés par une autre structure élémentaire de stockage de l'information : **qubit**. Un qubit, raccourci

de quantum bit, peut se définir comme l'état quantique qui représente la plus petite unité de stockage de l'information quantique. Il se compose de la superposition de deux états de base notés par convention $|0\rangle$ et $|1\rangle$. C'est une des différences majeures avec l'informatique conventionnelle qui ne peut prendre que les valeurs 0 et 1. Formellement, un qubit est souvent représenté par la notation de Dirac :

$$|\Psi\rangle = \alpha |0\rangle + b|1\rangle \text{ où } |\alpha|^2 + |b|^2 = 1$$

Formule 3.1. Notation de Dirac

Le tableau qui suit illustre un comparatif entre les bits classiques et les bits quantiques.

Bit classique	Bit quantique (qubit)
Un bit a toujours une valeur définie	Pas de valeur définie sans observation
Un bit vaut seulement 0 ou 1	Un qubit peut être dans une superposition de 0 et 1 simultanément.
Un bit peut être copié sans être affecté	Un qubit dans un état inconnu ne peut être copié
Un bit peut être lu sans affecter sa valeur	La lecture d'un qubit qui est initialement dans une superposition changera sa valeur
Lire un bit n'affecte pas un autre	Si un qubit est enchevêtré avec un autre, la lecture de l'un affectera le second.

Table 3.1. Bit vs Qubit [48]

3.2.2. Principes des calculateurs quantiques

Le principe d'un ordinateur quantique repose sur la manipulation d'un jeu de qubits. Chaque qubit peut porter soit un 0 soit un 1 soit une superposition des deux, une distribution de phase, qui prend la valeur 0 pour un angle de 90° , un 1 pour un angle de 0° et pour un angle entre les deux la superposition d'états dans les proportions du \sin^2 et du \cos^2 de la phase. L'interrogation d'un qubit qui n'a pas une phase p comprise entre 0° et 90° conduit à une réponse par 0 avec une probabilité de $\sin^2 p$ et à une réponse par 1 avec une probabilité de $\cos^2 p$ [52].

« Le secret de l'informatique quantique consiste donc à manipuler des objets de types électron ou photon et leur faire représenter une information non pas dans un état ou un autre mais plutôt dans tous les cas de figure possibles. Cette technique permet de multiplier de façon exponentielle l'information. Ainsi, un ordinateur à deux bits peut représenter un des quatre nombres suivants : 0,1, 10, 11, autrement dit 0, 1, 2 et 3 en langage binaire. Mais un ordinateur quantique de deux qubits est capable de stocker ces 4 nombres à la fois ! Un ordinateur de 300 qubits serait donc capable de stocker plus de bits qu'il n'existe d'atomes dans l'univers entier.

*Les ordinateurs quantiques servent donc à effectuer des calculs '**massivement parallèles**', mais avec une nuance : lorsque finalement la 'mesure' est prise, il ne reste plus de superposition et une seule des solutions devient alors disponible pour l'utilisateur. **C'est pourquoi on pense que les ordinateurs quantiques seront surtout utilisés dans des applications qui impliquent un choix parmi une multitude d'alternatives** » [46].*

Un algorithme quantique est un algorithme qui ne fonctionne que sur un ordinateur quantique. Il est défini comme une succession d'opérations quantiques. En règle générale, un algorithme quantique a une complexité moindre que les algorithmes équivalents qui s'exécutent sur des ordinateurs classiques grâce au concept quantique de la superposition. Parmi les algorithmes quantiques les plus célèbres, on cite celui de Shor (1994) pour la factorisation des nombres et celui de Grover (1996) pour la recherche dans une base de données non triée. Ces deux algorithmes ont résolu des problèmes dont la complexité a été réduite pour devenir linéaire.

3.2.3. Réalisation d'un ordinateur quantique (portes quantiques)

Les ordinateurs quantiques sont réalisés à base des portes quantiques, équivalentes des portes logiques en informatique classique. Une porte quantique est une transformation unitaire qui agit sur au plus 3 qubits. En réalité tout algorithme quantique doit être divisé en opérations élémentaires réalisées par des circuits quantiques. Avant de présenter comment un circuit quantique est réalisé à partir des portes quantiques, expliquons d'abord comment il est possible d'appliquer des opérations logiques sur des qubits. Le but de toute opération logique, est de changer la valeur des qubits d'une façon contrôlée.

Nous avons sélectionné trois portes quantiques s'appliquant à un, deux et trois qubits afin de cerner leur mode de fonctionnement.

- **Porte de Hadamard (lame demi-onde à $22,5^\circ$)** : Cette porte s'applique à un seul qubit. Elle permet une rotation de l'état d'un qubit, conduisant ainsi à créer une superposition. Les opérations impliquant un seul qubit sont en général simples.

$$|b\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle)$$

Figure 3.2. La porte quantique de Hadamard

- **Porte Non Contrôlée (CNOT)** : Elle s'applique à deux qubits. Son action est d'inverser le deuxième qubit si et seulement si le premier est dans l'état $|1\rangle$.

Entrée	Sortie
00	00
01	01
10	11
11	10

Table 3.2. Table de vérité de la porte CNOT

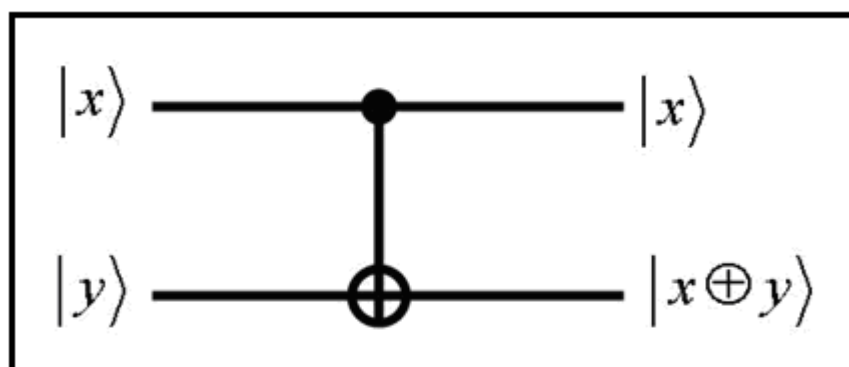


Figure 3.3. La porte quantique CNOT

- **Porte de Toffoli** : Elle s'applique à trois qubits. Son action est d'inverser le troisième qubit si et seulement si les deux premiers sont dans l'état $|1\rangle$.

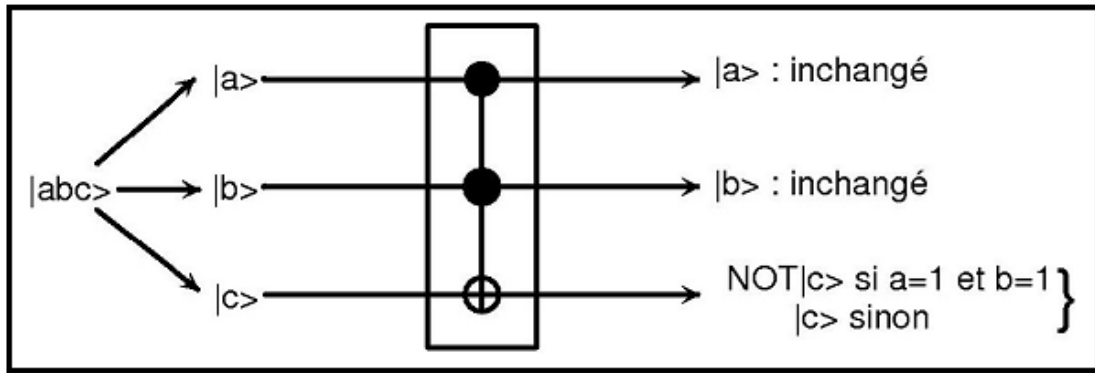


Figure 3.4. La porte quantique de Toffoli

- **Exemple d'un circuit quantique** : Un circuit quantique est une composition de portes quantiques.

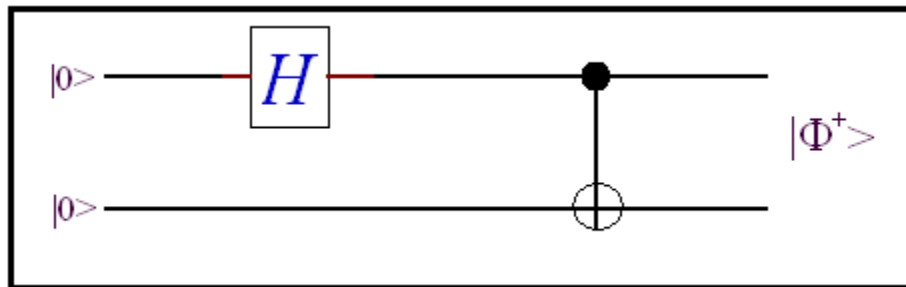


Figure 3.5. Exemple d'un circuit quantique [48]

Les opérations réalisées par ce circuit sont déroulées de la façon suivante :

1. $|0\rangle \otimes |0\rangle = |00\rangle$
2. $(U_H |0\rangle) \otimes |0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes |0\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |10\rangle)$
3. $\frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) = |\Phi^+\rangle$

3.3. Terminologie

Voici quelques concepts définissant le vocabulaire employé en informatique quantique [48] :

- **Superposition d'états** : Un qubit peut être dans l'état 0 ou 1 mais également les deux en même temps. Un registre de n qubits peut contenir 2^n valeurs distinctes. Le système travaille donc sur n nombres à la fois.

- **Interférence** : Il existe deux types d'interférences :
 - Constructive : Permet d'augmenter la probabilité d'obtenir un état.
 - Destructive : Permet de diminuer la probabilité d'obtenir un état.
- **Enchevêtrement** : Deux qubits sont enchevêtrés si l'un dépend totalement de l'autre.
- **Non déterminisme** : La mécanique quantique est non déterministe ce qui implique que l'informatique quantique l'exploite aussi. Le hasard quantique est authentique et irréductible, c'est une loi fondamentale.
- **Non duplication** : La copie n'est pas toujours possible car un état inconnu ne peut pas être dupliqué.
- **Le phénomène de décohérence** : C'est la perte des effets quantiques sur le long terme.
- **Intrication quantique** : L'intrication quantique est une base fondamentale en informatique quantique. En mécanique quantique, cela représente un état physique dans lequel deux systèmes $S1$ et $S2$ forment un seul et unique ensemble, quelque soit la distance les séparant.

4. Algorithmes génétiques quantiques (AGQ)

Un AGQ n'est rien d'autre qu'un AG dont les individus manipulés sont des chromosomes quantiques, c'est-à-dire une représentation basée sur le concept du qubit, et doté d'autres opérations quantiques. Ceci signifie que les opérations génétiques seront totalement redéfinies afin de pouvoir s'adapter avec la nouvelle représentation des chromosomes. La figure 3.6 illustre le fonctionnement d'un AGQ :

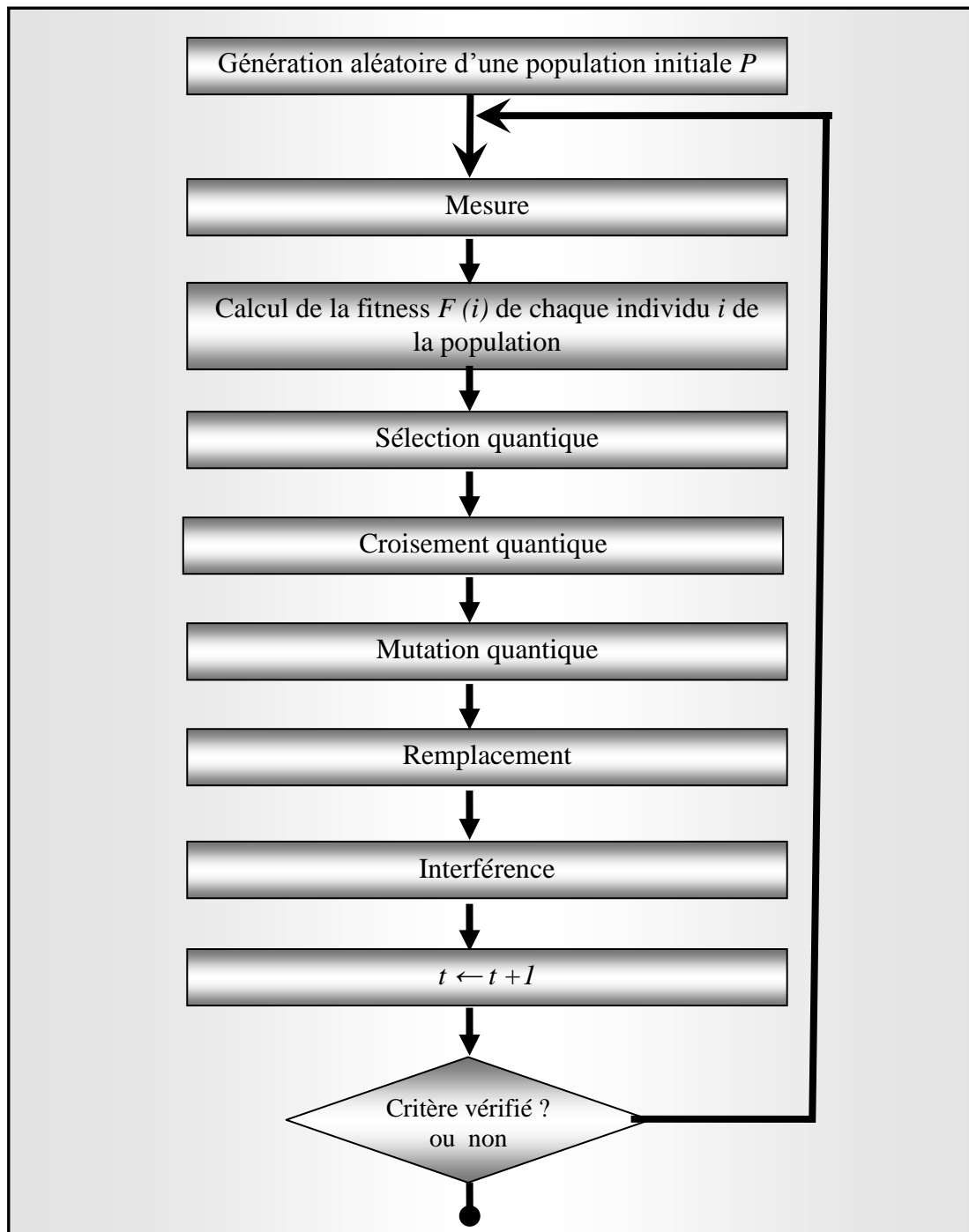


Figure 3.6. Principe de fonctionnement d'un AGQ

4.1. Codage des chromosomes quantiques

Comme l'élément de base ici est le qubit, un chromosome est tout simplement une chaîne de m qubits formant un registre quantique. La figure 3.7 montre la structure d'un chromosome quantique.

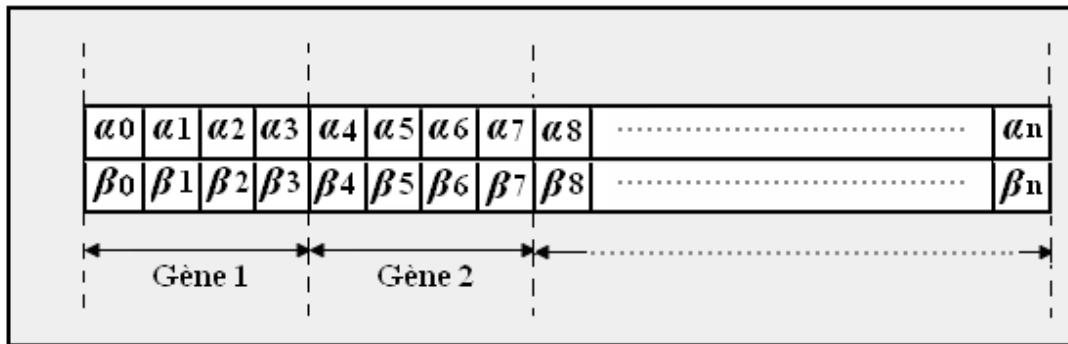


Figure 3.7. Structure d'un chromosome quantique.

4.2. Initialisation de la population

La génération de la population initiale a pour but de créer les chromosomes quantiques de la population initiale. La façon la plus simple est d'initialiser toutes les amplitudes des qubits par la valeur $2^{-1/2}$. Cela signifie qu'un chromosome quantique représente tous les états de superpositions avec la même probabilité.

4.3. Evaluation des individus

Le rôle de cette phase est similaire à celle d'un AG ordinaire : quantifier la qualité de chaque chromosome quantique dans la population pour effectuer une reproduction. L'évaluation se fait selon une fonction d'adaptation qui associe à chaque individu, *après sa mesure*, une valeur d'adaptation, et elle permet donc de noter les individus de la population.

4.4. Opérations génétiques quantiques

- **La fonction mesure :** Pour pouvoir exploiter efficacement les états superposés des qubits, il faut effectuer une opération de lecture pour chaque qubit. Contrairement à ce qui se passe dans un système quantique ordinaire, ici les états quantiques ne seront pas détruits permettant par là de conserver la superposition. Cette opération conduit à extraire un chromosome classique à partir d'un autre quantique. Le but étant de permettre l'évaluation des individus de la population en fonction des chromosomes classiques extraits.

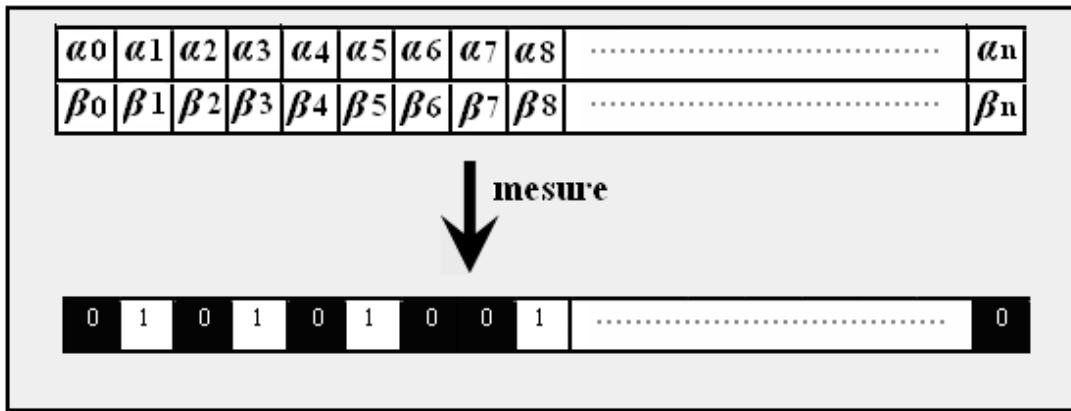


Figure 3.8. Mesure d'un chromosome quantique.

Une façon très simple pour implémenter cette fonction est donnée par le pseudo code qui se trouve dans la table 3.3.

```

Soient  $q$  un qubit et 'mesure' sa fonction de mesure.
 $q$  est défini par  $|\Psi\rangle = \alpha|0\rangle + b|1\rangle$  où  $|\alpha|^2 + |b|^2 = 1$ 

fonction mesure ()
{
     $r = \text{tirer } r \text{ dans } [0,1]$ 
    si ( $r > \alpha^2$ )
        retourner 1
    sinon
        retourner 0
}

```

Table 3.3. Implémentation de la fonction mesure

- **La sélection quantique :** Une fois réalisée l'évaluation de la génération, on opère une sélection à partir des valeurs d'adaptation. Seuls les individus passant l'épreuve de sélection peuvent se reproduire. Les types et les techniques de sélection restent en principe les mêmes que ceux d'un AG ordinaire.
- **Le croisement quantique :** Cette opération est similaire à celle d'un AG ordinaire sauf qu'elle opère sur les qubits d'un chromosome quantique. Les individus sélectionnés seront aléatoirement répartis en couples puis commencent à se reproduire. L'opération se fait par l'échange des fragments situés après les points de coupures ce qui permet une création de deux nouveaux chromosomes quantiques. Pratiquement, on peut réaliser un croisement en un point, deux points ou encore en plusieurs points.

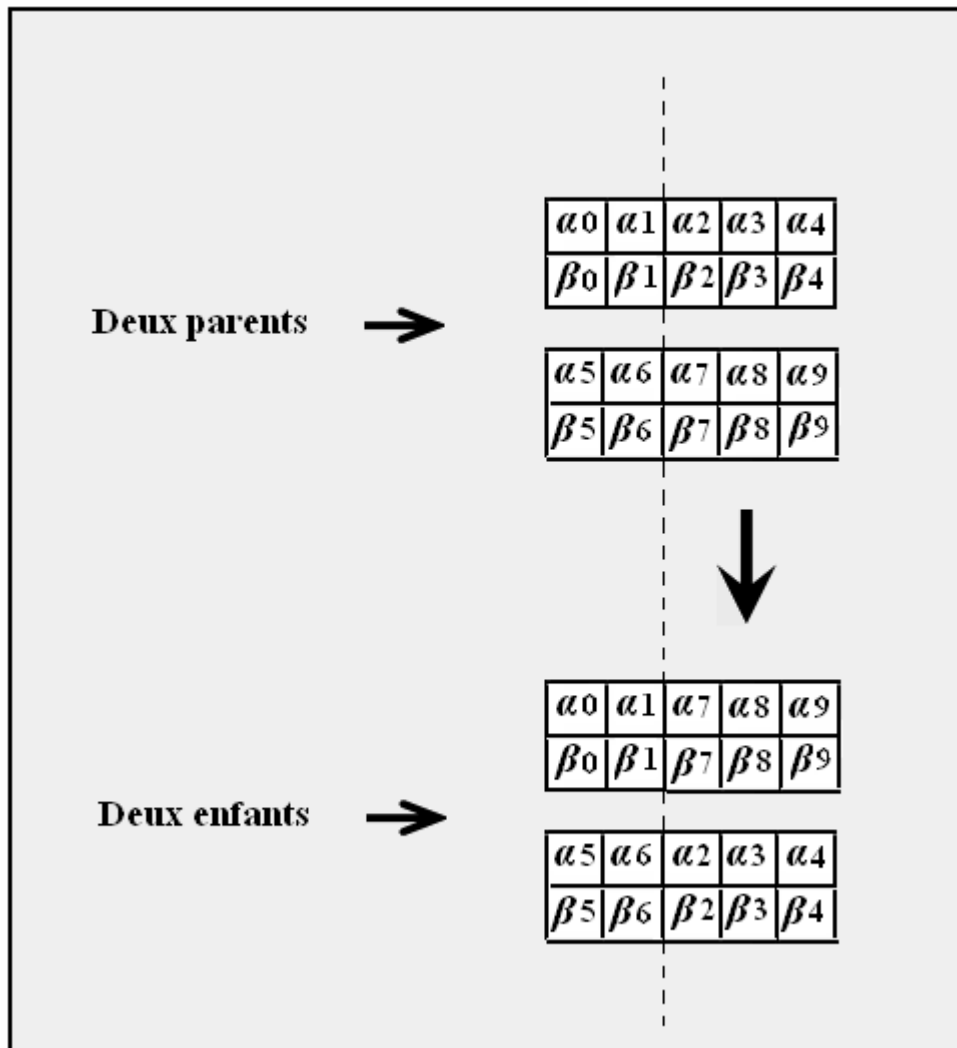


Figure 3.9. Croisement quantique en un point.

- **La mutation quantique :** Elle sert à changer la valeur de quelques positions aléatoires du chromosome quantique selon un taux de mutation. Ce changement se fait en permutant les amplitudes de la position mutée ou bien en les réinitialisant par la valeur $2^{-1/2}$.

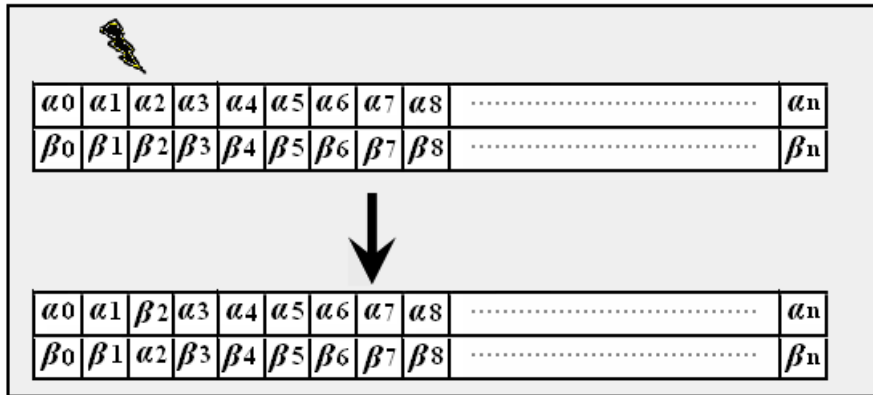


Figure 3.10. Mutation quantique.

- L'interférence :** Cette opération permet la modification des amplitudes des individus afin d'améliorer le rendement. Elle consiste principalement en le déplacement de l'état de chaque qubit dans le sens de la valeur de la meilleure solution trouvée. Cette opération est utile pour intensifier la recherche autour de la meilleure solution. Elle peut être réalisée à l'aide d'une transformation unitaire qui permet une rotation dont l'angle est une fonction des amplitudes (a_i, b_i) et de la valeur du bit correspondant à la solution référence. La valeur de l'angle de rotation $\delta\theta$ doit être choisie de manière à éviter la convergence prématurée. Elle est souvent fixée empiriquement et sa direction est déterminée en fonction des valeurs de a_i, b_i et de la valeur du qubit situant à la position i de l'individu en cours de modification [41].

5. Extension pour évoluer des AC

Nous avons vu dans le chapitre précédent (*cf II.3*) qu'il est possible de représenter un AC par un chromosome grâce à un simple système de conversion. Il semble être très simple de passer depuis un AC vers une structure d'un chromosome quantique en gardant le même système de conversion vu en (*cf II.3*). Cependant, au lieu de créer des sorties déterministes on génère des qubits jouant le rôle des états de sortie de l'AC. La détermination de l'automate est faite par un appel à la fonction « *mesure* » qui extrait un chromosome usuel à partir d'un autre quantique afin de permettre son évaluation. Cette dernière se fait d'abord en exécutant chaque chromosome extrait sur une ou un ensemble de configurations. Vient ensuite l'évaluation de l'ensemble de toutes les configurations finales à partir de laquelle il sera possible de déterminer la valeur d'adaptation de chaque chromosome.

6. Performances des AGQ

6.1. La présence du parallélisme

En 1966, Bernstein a introduit un ensemble de conditions permettant d'étudier la présence du parallélisme dans une application qui permet d'exécuter plusieurs tâches en parallèle. Soient deux tâches $T1$ & $T2$. On suppose que chacune d'entre elles utilise un ensemble de variables en entrée E et produit un ensemble de résultats en sortie S . On peut donc les représenter comme suit :

$$\begin{array}{ccc} T1 & \& T2 \\ E1 & & E2 \\ S1 & & S2 \end{array}$$

Selon Bernstein, $T1$ et $T2$ peuvent s'exécuter en parallèle si et seulement si les conditions suivantes sont satisfaites :

- $E1 \cap S2 = \emptyset$
- $E2 \cap S1 = \emptyset$
- $S1 \cap S2 = \emptyset$

Cette définition peut également être généralisée pour étudier la présence du parallélisme dans k tâches, dans ce cas les conditions de Bernstein doivent être respectées pour chaque couple $(Ti // Tj)$ avec $i \neq j$.

Donc, pour pouvoir exécuter deux tâches en parallèle, il faut qu'elles soient indépendantes et qu'il existe suffisamment de ressources pour les exécuter. Nous allons étudier dans ce qui suit la présence du parallélisme dans les AGQ.

6.2. Application des conditions de Bernstein sur les AGQ

L'un des points clés des AG est la possibilité d'effectuer des traitements parallèles. Dans un AGQ on distingue plusieurs tâches à exécuter : l'évaluation, la sélection, le croisement, la mutation, l'interférence et le remplacement. Il est évident que les conditions de Bernstein ne sont pas vérifiées entre ces tâches du fait que chacune d'entre elles dépend de celle qui la précède. Formellement, on écrit :

$$\forall i > 0 E_i \cap S_{i-1} \neq \emptyset$$

Formule 3.2. Application des conditions de Bernstein sur les tâches d'un AGQ

Nous pouvons donc déduire que si un tel algorithme présente du parallélisme, il faut le chercher au sein des tâches elles-mêmes.

6.2.1. L'évaluation

Cette tâche peut être divisée en tâches plus petites qui vérifient les conditions de Bernstein et donc pouvant s'exécuter en parallèle. Chaque sous tâche consiste à évaluer un individu de la population en lui affectant une valeur d'adaptation. On exprime ceci par les expressions de la formule 3.3.

$T_{\text{évaluation}} = \{ T_{\text{évaluation_individu}_i} / i= 1..n \}$ n étant la taille de la population.

$\forall i :$ $T_{\text{évaluation_individu}_i}$

$E_i :$ individu i

$S_i :$ valeur d'adaptation v_i

Formule 3.3. Décomposition de la tâche d'évaluation

On peut maintenant vérifier que les conditions de Bernstein sont respectées entre les sous tâches, et elles peuvent donc s'exécuter en parallèle.

6.2.2 La sélection

Contrairement à la tâche d'évaluation, cette tâche ne peut généralement se diviser en sous tâches car elle sert à trier un ensemble d'individus selon leurs valeurs d'adaptation puis sélectionner un sous ensemble pour la reproduction, nous allons donc la considérer comme une tâche ne pouvant s'exécuter en parallèle.

6.2.3 Le croisement

Cette tâche peut se diviser en deux sous tâches : une séquentielle et l'autre parallèle. La première sert à construire l'ensemble des couples d'individus voulant se reproduire et elle n'est donc pas divisible en d'autres sous tâches. La deuxième est l'opération de reproduction proprement dite. Cette dernière peut se décomposer en d'autres sous tâches pouvant s'exécuter en parallèle, ceci est exprimé par la formule 3.4.

$T_{reproduction} = \{ T_{reproduction_couple_i} / i = 1..m \}$ m étant le nombre de couples.

$\forall i : T_{reproduction_couple_i}$

$E_i : couple\ i$

$S_i : deux\ nouveaux\ individus$

Formule 3.4. Décomposition de la tâche de reproduction

Il est trivial que les conditions de Bernstein sont vérifiées et on peut donc effectuer un traitement parallèle.

6.2.4 La mutation

Cette tâche est similaire à celle de l'évaluation et elle peut donc s'exécuter en parallèle. Formellement, on écrit :

$T_{mutation} = \{ T_{mutation_individu_i} / i = 1..n \}$ n étant la taille de la population.

$\forall i : T_{mutation_individu_i}$

$E_i : individu\ i$

$S_i : individu\ i\ muté$

Formule 3.5. Décomposition de la tâche de mutation

On peut facilement vérifier que les conditions de Bernstein assurent l'exécution parallèle de cette tâche.

6.2.5. L'interférence

Cette tâche peut être divisée en tâches plus petites qui vérifient les conditions de Bernstein et elles peuvent donc s'exécuter en parallèle. Chaque sous tâche consiste à modifier les amplitudes d'un certain nombre d'individus de la population en fonction du meilleur individu. Formellement, on exprime ceci par la formule 3.6.

$T_{interférence} = \{ T_{interférence_individu_i} / i= 1..m \}$ m étant le nombre d'individus interférés.

$\forall i : T_{interférence_individu_i}$

$E_i : individu\ i$

$S_i : individu\ i\ interféré$

Formule 3.6. Décomposition de la tâche d'interférence

On peut facilement vérifier que les conditions de Bernstein sont satisfaites et on peut donc effectuer une exécution parallèle.

6.2.6. Le remplacement

De manière générale, il est préféré de considérer cette tâche comme séquentielle car on ne sait pas comment les individus seront remplacés (politique de remplacement). En outre, ceci peut être fortement lié aux choix techniques d'implémentation.

6.3. Les limites du parallélisme

6.3.1. La loi d'Amdahl

Elle indique la limite de l'accélération atteignable par une architecture parallèle pour l'exécution d'un programme particulier. Elle définit l'accélération pouvant être obtenue à l'aide d'une caractéristique particulière.

$A = \frac{\text{temps d'exécution pour la tâche entière sans utiliser la caractéristique}}{\text{Temps d'exécution pour la tâche en utilisant la caractéristique}}$

Formule 3.7. La loi d'Amdahl

6.3.2. Interprétation

Soit X un programme composé d'une partie P exécutable en parallèle et d'une partie S ne pouvant être exécutée en parallèle. On note par p le temps d'exécution séquentielle de la partie P et par s le temps d'exécution de la partie S . Soit n le nombre de processeurs utilisés pour l'exécution parallèle de la partie P . L'accélération sera donc réécrite comme indiquée par la formule 3.8.

$A = (s + p) / (s + p/n)$

Formule 3.8. Interprétation de la loi d'Amdahl

Si on met $s = \gamma (s + p)$ avec γ en pourcentage (%), on aura aussi $p = (1 - \gamma) (s + p)$.
L'accélération A devient donc :

$$A = 1 / (\gamma + (1 - \gamma) / n)$$

Formule 3.9. Réécriture de la loi d'Amdahl

Plus γ est proche de 1 plus l'accélération tend vers 1, mais si γ est plus proche de 0 l'accélération tend vers le nombre de processeurs n . On peut donc conclure à partir de cette loi que quand il est demandé d'écrire un programme parallèle il faut limiter autant que possible sa partie séquentielle.

Que peut-on dire sur les performances d'un AGQ ? D'après les conditions de Bernstein, il semble qu'un tel algorithme a une grande partie pouvant s'exécuter en parallèle. En revanche, il existe aussi une autre partie considérable et coûteuse ne pouvant s'exécuter que séquentiellement. Prenons l'exemple de la tâche de sélection, pour pouvoir sélectionner des individus, on doit d'abord effectuer un algorithme de tri similaire à ceux concernant les tableaux unidimensionnels ce qui peut entraver l'exécution parallèle et donc ralentir les performances du système de façon remarquable. Une conclusion est que les AGQ présentent un parallélisme handicapé par des parties séquentielles influençant leurs performances globales.

Notons par p_{ev} , s_{select} , $\{s_{crois} + p_{crois}\}$, p_{mut} , p_{interf} , s_{remp} , les temps d'exécutions séquentielles des tâches d'évaluation, de sélection, de croisement, de mutation, d'interférence et de remplacement respectivement. L'application de la loi d'Amdahl est donnée par la formule 3.10.

$$A = (s_{select} + s_{crois} + s_{remp} + p_{ev} + p_{crois} + p_{mut} + p_{interf}) / (s_{select} + s_{crois} + s_{remp} + (p_{ev} + p_{crois} + p_{mut} + p_{interf}) / n)$$

Formule 3.10. Application de la loi d'Amdahl sur un AGQ

On peut maintenant juger qu'il est probable que la valeur prise par γ dans la formule d'Amdahl peut ne pas être comme l'on désire. Est-il possible de dépasser ces limites ?

6.3.3. La loi de Gustavson

Cette loi modère la conclusion de la loi d'Amdahl. Soit X un programme composé d'une partie P exécutable en parallèle et d'une partie S ne pouvant être exécutée en parallèle. p étant le temps d'exécution séquentielle de la partie P et s est le temps d'exécution de la partie S . Soit n le nombre de processeurs utilisés pour l'exécution parallèle de la partie P . Selon la loi d'Amdahl, le temps de calcul est donné par : $t = s + p/n$.

Gustavson a supposé que la partie parallèle augmente linéairement avec n , nombre de processeurs, on écrit : $p = an$, le temps d'exécution devient donc : $t = s + a$. L'accélération sera autrement réécrite : $A = (s + an) / (s + a)$. Que se passe-t-il si " $a \rightarrow \infty$ " ? A tendra vers le nombre de processeurs n .

Une conclusion est que plus la taille a de la partie attribuée à chaque processeur est importante plus l'accélération tend vers le nombre de processeurs. Ceci implique qu'il est possible que le parallélisme dépassera la limite établie par Amdahl à condition d'augmenter la quantité d'informations traitées par chaque processeur. En pratique, nos expérimentations sur l'évolution des AC par des AE ont montré que la plupart du temps d'exécution est écoulé pendant la phase d'évaluation, autrement dit : $p_{ev} \gg (s_{select} + s_{crois} + s_{rem})$
ce qui permet une exécution parallèle à haute performance.

7. Algorithme proposé basé sur les AGQ pour l'évolution des AC

Dans cette section nous allons proposer un AGQ pour l'évolution des AC. L'originalité de cet algorithme tient à ce que l'analyse des travaux effectués sur ce domaine, comme présentée dans l'état de l'art du premier chapitre, a montré qu'aucun chercheur n'a abordé l'évolution des AC par des AGQ. C'est ce manque qui nous a encouragés à s'engager dans cette voie. Nous avons ainsi conçu cet algorithme que nous avons appelé par : **automate cellulaire évolutionnaire quantique (ACEVQ)**.

L'algorithme proposé ici fonctionne sans avoir recourt ni à une sélection ni à un croisement ni même à une mutation. Cela veut dire que l'interférence est le seul opérateur génétique à appliquer. En effet, la mutation et le croisement peuvent faire changer les probabilités de la superposition linéaire des états quantiques. Mais, comme dans les AGQ la diversité génétique est principalement causée par la représentation du qubit, il n'est pas si nécessaire d'utiliser les autres opérateurs génétiques. Si par exemple les taux de mutation et de croisement sont élevés, les performances de l'AGQ peuvent être diminuées. Nous avons cherché à réduire le

nombre des paramètres pour rendre l'algorithme plus facile à utiliser. Cette stratégie est un peu similaire, **mais pas la même**, à celle adoptée par [31] où son algorithme, qui était utilisé pour résoudre le problème du sac à dos, n'a utilisé que l'interférence comme opérateur génétique. La figure 3.11 illustre son fonctionnement :

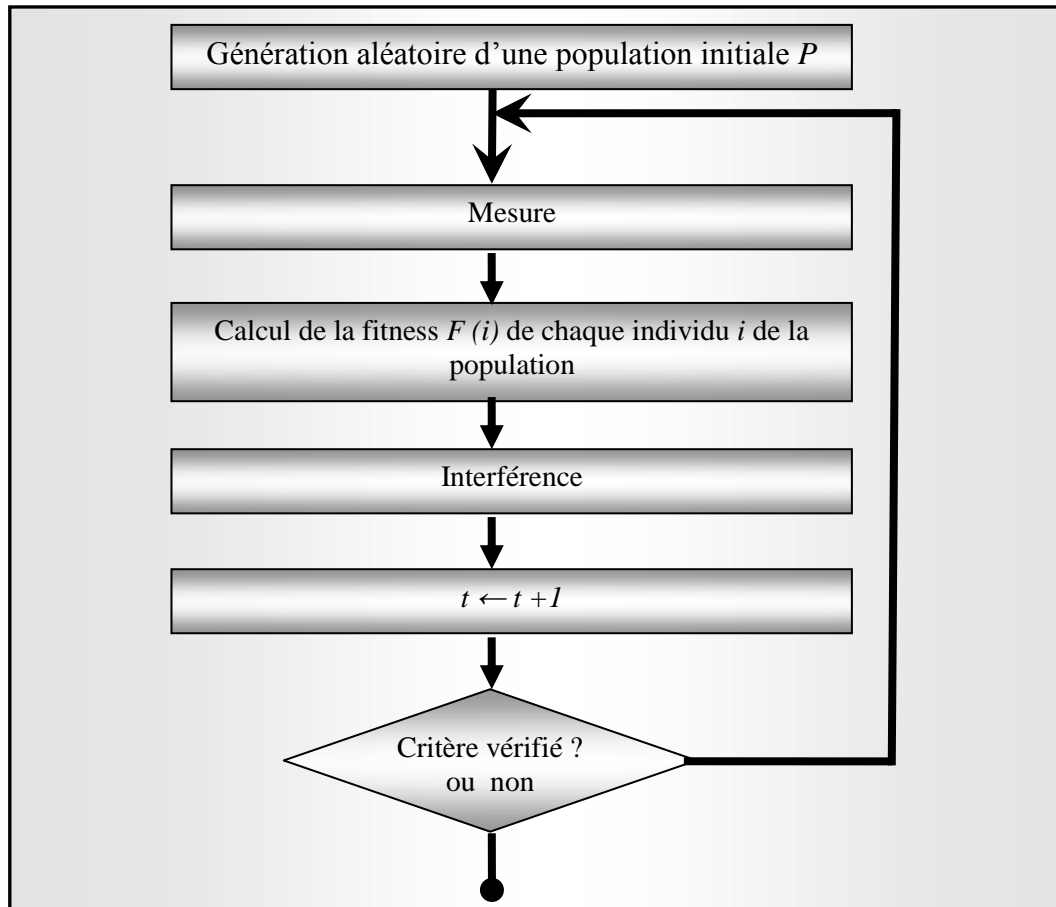


Figure 3.11. Principe de fonctionnement de l'AGQ de [31]

- **Principe de fonctionnement de notre algorithme :**

La rotation des amplitudes des individus est faite par des portes quantiques. La porte quantique peut également être conçue en conformité avec le problème présent. Bien que de nombreuses portes quantiques puissent être sélectionnées, la rotation des portes quantiques est préférable en raison du caractère du calcul des AGQ. La population $P(t)$ est donc mise à jour avec une rotation des portes quantiques. Notre politique de rotation est donnée par la formule 3.11.

$$\begin{bmatrix} \alpha_i^{t+1} \\ \beta_i^{t+1} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i^t \\ \beta_i^t \end{bmatrix}$$

Formule 3.11. Politique de rotation des qubits

Où $\Delta\theta_i$ est l'angle de rotation de la porte quantique du qubit i d'un individu. Elle est souvent obtenue à partir d'un tableau de recherche afin d'assurer la convergence. La table 3.4 représente notre stratégie de convergence.

x_i	b_i	$f(x) > f(b)$	$\Delta\theta_i$	$s(a_i b_i)$			
				$a_i b_i > 0$	$a_i b_i < 0$	$a_i = 0$	$b_i = 0$
0	0	0	$\Delta\theta_2$	-	+	\pm	\pm
0	0	1	$\Delta\theta_2$	-	+	\pm	\pm
0	1	0	$\Delta\theta_1$	-	+	\pm	\pm
0	1	1	$\Delta\theta_2$	-	+	\pm	\pm
1	0	0	$\Delta\theta_1$	+	-	\pm	\pm
1	0	1	$\Delta\theta_2$	+	-	\pm	\pm
1	1	0	$\Delta\theta_2$	+	-	\pm	\pm
1	1	1	$\Delta\theta_2$	+	-	\pm	\pm

Table 3.4. Table de recherche pour la rotation des portes quantiques

x_i et b_i sont les i èmes bits de x et b (la meilleure solution trouvée) respectivement. f est la fonction fitness et $s(a_i b_i)$ est le signe de l'angle θ_i . D'après la table de recherche, on peut facilement constater que cette stratégie améliore, pour chaque individu, les amplitudes des qubits qui sont mauvaises selon un angle $\delta\theta_1$ tandis qu'elle diminue celles qui sont bonnes selon un angle $\delta\theta_2$. La modification des amplitudes des qubits se fait selon les signes des amplitudes, la meilleure solution trouvée et la solution extraite par l'individu la contenant. Il est normal que $\delta\theta_1$ soit $\gg \delta\theta_2$ car la diminution des amplitudes sert à corriger seulement les erreurs stochastiques permettant d'une part d'éviter la dérive génétique et de garantir la diversité génétique de l'autre part. Nous allons annoncer dans le chapitre qui suit les valeurs des paramètres utilisés ici comme la taille de la population et l'angle de rotation ainsi qu'un ensemble de résultats expérimentaux.

8. Conclusion

Nous avons vu dans le chapitre précédent que la difficulté de concevoir un automate cellulaire pour présenter un comportement spécifique exige l'automatisation du processus de la conception : explorer des espaces d'AC par des AG. La technique des AG s'est avérée efficace pendant plusieurs décennies en raison de nombreux avantages dont les plus importants sont le parallélisme et la généralité.

Nous avons vu dans ce chapitre que les recherches effectuées sur le domaine de l'informatique quantique ont récemment connu un intérêt croissant et ce vu les insuffisances des ordinateurs actuels. De nombreux problèmes d'optimisation ont par conséquent été résolus en appliquant l'approche des AGQ. A notre tour, nous avons étendu le concept d'ACEV en un autre que l'on a appelé *automate cellulaire évolutionnaire quantique (ACEVQ)* où les AC ont été évolués par des AGQ en les modélisant par des chromosomes quantiques. D'après leur corps algorithmique, il semble que les AGQ sont relativement performants, cela a été mis en exergue par des études théoriques prouvant leur puissance d'exécution parallèle. Il ne reste donc qu'à les tester pour savoir jusqu'à quelle mesure peuvent-ils s'avérer efficaces et performants sachant qu'ils sont fondés sur un mécanisme stochastique. La réponse à cette question fait l'objet d'une discussion d'un ensemble de résultats expérimentaux, concrétisée par le chapitre suivant.

*Parallélisation
des automates
cellulaires
évolutionnaires
quantiques*

Chapitre

4

19. Introduction
20. Plateforme pour la simulation des AC
21. Evolution des AC par des AGQ
22. Application des ACEVQ sur divers problèmes
23. Résultats expérimentaux
24. Conclusion

Parallélisation des automates cellulaires évolutionnaires quantiques

1. Introduction

Nous avons vu dans le chapitre précédent que les ACEVQ peuvent théoriquement supporter des traitements parallèles. Cependant, ni leur efficacité ni leur performance n'ont été mesurées. Ce chapitre est donc consacré à l'étude de ces deux propriétés. La première a été mesurée en appliquant les ACEVQ sur deux problèmes différents : le problème de la classification de densité $\rho_c = 1/2$ et celui des bitmaps. La deuxième a été mesurée en prenant appui sur un réseau local d'ordinateur en étendant le concept d'ACEVQ en un autre que l'on a appelé par **automate cellulaire évolutionnaire quantique parallèle (ACEVQP)**.

Nous commençons tout d'abord par la présentation de notre plateforme de simulation des AC qui consiste essentiellement en un ensemble de classes Java doté d'une API permettant sa réutilisation. Ensuite, nous présentons la version parallèle des ACEVQ ainsi que son implémentation. Enfin, nous présentons une série d'expérimentations ainsi qu'un ensemble de résultats expérimentaux afin de montrer la puissance de cette approche.

2. Plateforme pour la simulation des AC

Vu le faible nombre d'outils de simulation développés au sein de notre laboratoire, il était utile de concevoir notre propre outil de simulation. Nous avons ainsi conçu une plateforme développée en Java capable de simuler des AC pouvant théoriquement s'exécuter sur des grilles de n'importe quelle dimension, elle est également dotée d'un mécanisme pour supporter l'évolution des AC par des AG et des AGQ.

2.1. Idée de départ

Habituellement, on trouve que dans la plupart des plateformes destinées à la simulation des AC, les traitements concernant le calcul des voisins d'une cellule ne sont pas séparés de la topologie du réseau cellulaire (CAV et EvoCell sont par exemple deux plateformes qui ne supportent que la manipulation des AC à 2D), la difficulté est donc principalement due à l'implémentation de la fonction calculant le voisinage des cellules qui dépend de la topologie de l'AC.

- **Exemple**

Soit A un AC **bidimensionnel** évoluant une configuration de cellules C . V étant l'ensemble des états cellulaires et r est le rayon du voisinage.

Si par exemple on adopte le voisinage de *Moore* l'ensemble des voisins d'une cellule sera donné par la formule 4.1.

$$N_{ij} = \{C(k, l) \in V // |k-i| \leq r \text{ et } |l-j| \leq r\}$$

i, j sont les coordonnées de la cellule dans la grille

Formule 4.1. Calcul du voisinage de Moore [AC à 2-D]

Par contre si l'on adopte un voisinage de *Neumann* l'ensemble des voisins sera calculé par la formule 4.2.

$$N_{ij} = \{C(k, l) \in V // |k-i| + |l-j| \leq r\}$$

i, j sont les coordonnées de la cellule dans la grille

Formule 4.2. Calcul du voisinage de Neumann [AC à 2-D]

Maintenant, supposons que nous désirions opérer cela pour un AC **unidimensionnel**, il est clair que les formules de calcul seront carrément changées car la topologie de l'automate l'exige. Les formules de calcul des voisinages deviennent donc :

- **Voisinage de Moore**

$$N_i = \{C(k) \in V \mid |k-i| \leq r\}$$

i est la coordonnée de la cellule dans la grille

Formule 4.3. Calcul du voisinage de Moore [AC à 1-D]

- **Voisinage de Neumann**

$$N_i = \{C(k) \in V \mid |k-i| \leq r\}$$

i est la coordonnée de la cellule dans la grille

Formule 4.4. Calcul du voisinage de Neumann [AC à 1-D]

Par analogie, lorsqu'on est en présence d'un automate ayant N dimensions les formules de calcul seront également changées. Pour surmonter ce problème, nous avons construit un système dont la fonction de calcul des voisinages est totalement indépendante de la topologie du réseau cellulaire, ce qui permet d'évoluer plusieurs types d'AC sans changer la définition des opérations de calcul, **l'idée repose sur l'utilisation d'un espace cellulaire virtuel.**

2.2. Transformation des configurations

Notre système détecte automatiquement la topologie du réseau en déduisant sa dimension D . Une fois la dimension D déterminée, la configuration sera transformée en une autre dite *virtuelle* en appliquant un système de conversion. L'espace virtuel des cellules est un **espace unidimensionnel** où à chaque cellule dont la position dans la configuration originale est de $(x_1, x_2, x_3, \dots, x_d)$ correspond une cellule dont la position est de i , calculée par la formule 4.5.

$$i = x_1 * D_1 + x_2 * D_2 + x_3 * D_3 + \dots + x_d * D_d$$

Formule 4.5. Calcul des indexes des configurations virtuelles

On peut alors dire qu'on est passé à *un niveau d'abstraction plus élevé* où les configurations de cellules seront considérées comme **des ensembles de cellules ayant chacune une position donnée**. Contrairement à la topologie des automates unidimensionnels, ici une cellule dont sa position est de i n'appartient pas forcément au voisinage de celle dont sa position est de $(i - r)$ ou de celle dont sa position est de $(i + r)$.

La réalisation de cette correspondance est très complexe du fait qu'on ne connaît pas la dimension des configurations a priori. Nous avons vaincu cette complexité grâce à un ensemble de fonctions récursives.

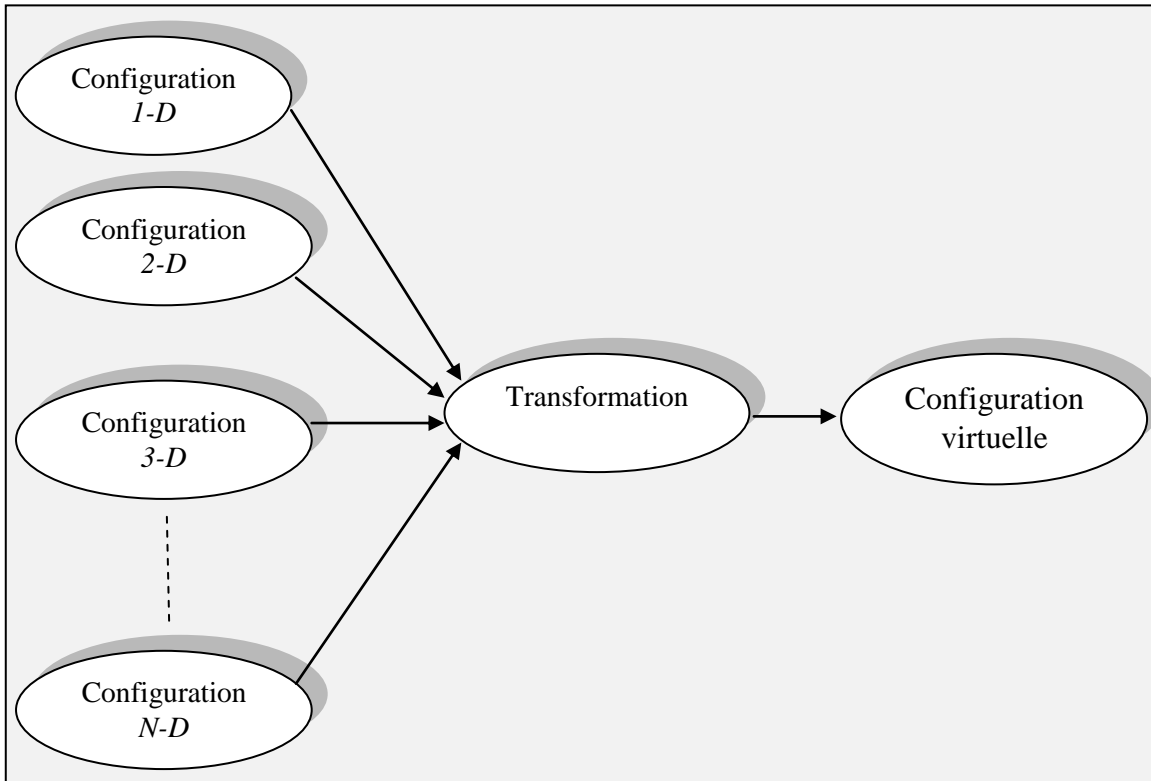


Figure 4.1. Système de transformation des configurations

2.3. Architecture de la plateforme

La plateforme développée est composée de trois couches, chacune consiste en un ensemble de *classes Java*. Chaque couche utilise directement les classes de la couche inférieure. La figure 4.2 illustre sa représentation graphique.

API d'utilisation					
Couche haute	Chromosome	Evolution par AG	Evolution par AGQ	HereBoy	SFFS
Couche médiane	Règle de Transition	Configuration		Visualisation	
Couche basse	Cellule	Model de Cellules	Voisinage	Processeur de Données	

Figure 4.2. Architecture d'une plateforme pour des AC à N dimensions

Nous allons maintenant détailler le rôle de chaque composant afin d'illustrer la relation entre les différentes couches.

2.3.1. La couche basse (Low Level)

Cette couche est principalement liée à l'aspect selon lequel les cellules d'un automate sont modélisées et gérées. Elle sert à définir l'ensemble des états cellulaires possibles ainsi que les règles de voisinages qui gèrent le réseau cellulaire. Elle est donc une couche préparatoire pour la couche supérieure. Elle est essentiellement composée de :

- **La classe Cellule (Cell)** : Cette classe représente une abstraction d'une cellule. Elle est caractérisée par un état qui est choisi d'être simplement un entier.
- **La classe Modèle de Cellules (Cells Model)** : Cette classe permet de définir l'ensemble des états cellulaires, elle représente l'ensemble V étudié en (cf. II.1.2). Elle permet aussi d'attribuer à chaque état cellulaire une couleur, essentiellement pour des raisons de visualisation.
- **La classe Voisinage (Neighborhood)** : Cette classe permet de définir le rayon r de l'environnement d'une cellule dans l'espace cellulaire, le voisinage d'une cellule est par conséquent calculé à partir d'un rayon choisi.
- **La classe Processeur de Données (Data Processing)** : C'est la classe la plus importante et la plus complexe parmi celles qui constituent cette couche. Nous allons étudier son rôle lors de l'étude de la classe *Configuration* de la couche suivante.

2.3.2. La couche médiane (Medium Level)

Cette couche est responsable de l'évolution d'un AC donné. Elle permet la définition de la configuration des cellules à évoluer et la fonction de transition qui permet la détermination de l'état futur de chaque cellule. Elle est principalement composée de trois classes définies comme suit :

- **La classe Configuration** : Nous avons vu dans le chapitre II (cf. II.1.2) que le réseau cellulaire peut prendre corps dans des espaces à D dimensions. Vu cette diversité, notre système est développé de façon à ce que toutes les topologies seront transformées en des configurations virtuelles. Les données de chaque configuration réelle demeurent telles qu'elles sont, c'est-à-dire sa structure reste sans modification.

Par conséquent, sa gestion se fait en gérant virtuellement des indexes pour rendre toutes les cellules accessibles.

L'évolution aura donc lieu en parcourant la table virtuelle case par case tout en passant l'index de chaque case à la classe "*Processeur de Données*" qui le convertit en un index exprimé sous une forme compréhensible par la topologie réelle de l'automate. Une fois cette transformation faite, il ne reste plus qu'à extraire les voisins de cette cellule en retournant un tableau de voisinage permettant la déduction de son état prochain par la fonction de transition.

La classe "*Processeur de Données*" est alors le cœur de cette plateforme car elle est le responsable de la définition des voisinages et de la conversion des indexes tandis que la classe *Configuration* n'est rien d'autre que son abstraction à un niveau plus élevé permettant ainsi sa meilleure exploitation.

- **La classe Règle de Transition (CA Rule) :** Cette classe est une abstraction d'une fonction de transition d'un AC, elle contient une méthode abstraite appelée '*getOutputOf*' qui retourne l'état futur d'une cellule à partir de son tableau du voisinage. En effet, cette méthode doit être abstraite car il peut y avoir plusieurs types de fonctions de transitions : extensionnelles et intentionnelles. Les premières servent à représenter une fonction de transition sous forme d'un graphe de V^n dans V et donc faire associer pour chaque voisinage possible l'état futur de la cellule concernée. Les secondes servent à la représenter sous forme de métarègles ayant la forme "*si ... alors*". Nous avons alors distingué deux types de fonctions de transitions, chacune d'entre elles peut à son tour comporter plusieurs variantes. Pour les fonctions extensionnelles, on peut trouver celles prenant en compte les conditions de la symétrie par rapport aux rotations 90° , 180° , 270° et le retournement flip-flop horizontal et vertical, et celles ne les prenant pas en compte. Pour les fonctions intentionnelles on distingue celles adoptant un voisinage aveugle et celles ne l'adoptant.

Pour chacune des fonctions de transitions précitées est associée une classe qui hérite la classe *Règle de Transition*. Pour les classes intentionnelles, les règles définissant les fonctions de transitions sont construites à partir d'un fichier *XML* contenant la configuration appropriée à leur fonctionnement.

- **La classe Visualisation (Configuration Display) :** Cette classe est spécialement utilisée pour suivre graphiquement l'évolution de la configuration selon le modèle de cellules utilisé. Seuls les automates unidimensionnels et les automates bidimensionnels sont supportés par cette classe. Pour les AC unidimensionnels, elle sert à visualiser les diagrammes espace-temps et pour ceux bidimensionnels elle sert à visualiser l'évolution de la configuration à chaque pas de temps.

2.3.3. La couche haute (Hight Level)

Cette couche est responsable de l'évolution des AC par d'autres algorithmes comme les AG par exemple. Elle est principalement basée sur la structure du chromosome pour représenter des fonctions de transitions et ce en raison de ses nombreux avantages. Elle est essentiellement composée de :

- **La classe Chromosome (Chromosom) :** Comme dans les AG, un chromosome est une suite de gènes dont chacun possède un locus. Partant de cette définition, on peut facilement déduire que cette classe doit forcément hériter une classe représentant une règle de transition extensionnelle en la munissant par des opérations génétiques (mutation, croisement). Cette classe permet facilement de passer vers des structures de chromosomes quantiques.
- **La classe Evolution Par AG (GA Evolution) :** Cette classe est responsable de l'évolution d'une population de chromosomes sur un ensemble de configurations selon le principe des AG (évaluation, sélection, croisement, mutation et remplacement). Les parties pouvant s'exécuter en parallèle sont implémentées par des *Threads ou tâches d'exécutions* permettant ainsi leur exécution **en concurrence**. L'utilisateur doit également, définir selon le besoin, les valeurs de quelques paramètres comme le taux de mutation et le pourcentage des individus sélectionnés.
- **La classe Evolution Par AGQ (QGA Evolution) :** Cette classe est responsable de l'évolution d'une population de chromosomes quantiques selon le principe des AGQ (mesure, évaluation, interférence). Les parties pouvant s'exécuter en parallèle sont aussi implémentées par des *Threads*. L'utilisateur doit définir les valeurs de quelques paramètres comme l'angle de rotation et la taille de la population.

- **La classe HereBoy** : Cette classe est une implémentation de l'algorithme *HereBoy* qui est un algorithme combinant les techniques des AG avec celles des recuits simulés. Cet algorithme n'utilise qu'un seul chromosome et une mutation adaptative pour évoluer un ensemble de configurations. L'utilisateur doit donc définir les valeurs des paramètres de l'algorithme, comme le taux de mutation, selon le type de problème à résoudre.
- **La classe SFFS** : Cette classe est une implémentation de l'algorithme *SFFS* (Séquentiel Floating Forward Search) qui est un algorithme déterministe pour la sélection des fonctionnalités pour la construction des systèmes classificateurs. Cet algorithme peut être étendu pour supporter l'évolution des AC [38]. Son implémentation a nécessité l'utilisation d'un seul chromosome pour évoluer un ensemble de configurations. Contrairement aux algorithmes précédents, celui-ci n'a qu'un seul paramètre à définir : le nombre maximum d'itérations car c'est un algorithme déterministe, c'est-à-dire les mêmes entrées conduisent toujours à reproduire les mêmes sorties.

Notre objectif au futur est de construire une version améliorée qui supporte des traitements parallèles et distribués via des objets distribués (comme RMI, Remote Method Invocation, par exemple) garantissant une exécution parallèle et distribuée, éventuellement sur des grilles de calculs.

3. Adaptation des ACEVQ pour supporter un traitement parallèle

Dans cette section nous allons présenter la version parallèle des ACEVQ. **Le but principal est d'assurer une haute performance d'exécution de l'algorithme proposé.** Nous allons également présenter les choix techniques d'implémentation de notre algorithme. Nos résultats expérimentaux seront présentés par la suite dans la section 5.

3.1. Parallélisation de l'algorithme proposé

Nous avons vu dans le chapitre précédent que les tâches d'évaluation et d'interférence peuvent s'exécuter en parallèle. Il est donc possible s'il existe plusieurs processeurs, comme dans un réseau local d'ordinateurs par exemple, d'évaluer et d'interférer plusieurs chromosomes en même temps. Dans ce cas il est clair qu'après avoir effectué une évaluation

des individus, les processeurs devront être synchronisés afin de se mettre d'accord sur la valeur de la meilleure solution trouvée qui sera utilisée lors de la phase d'interférence.

En respectant cette exigence de synchronisation entre les processeurs, nous avons construit une version parallèle de l'algorithme vu en (cf. III.7) selon le modèle **maitre / esclave** en s'appuyant sur un réseau local d'ordinateurs *LAN*. Son principe de fonctionnement est le suivant :

Soit un réseau local de $m + 1$ sites.

- On choisit arbitrairement un site i qui joue le rôle d'un coordinateur pour la synchronisation des sites exécutant l'opération d'évaluation. Chaque site terminant l'exécution de l'opération d'évaluation doit informer le site i qui réagit par la détermination de la meilleure solution puis l'envoyer aux différents sites pour réaliser l'opération d'interférence en parallèle.
- Les m sites restants sont donc des esclaves chargés de réaliser les tâches d'évaluation et d'interférence de manière parallèle en exécutant les ordres du site maitre.

La figure 4.3 illustre la structure du modèle d'exécution parallèle maitre / esclave.

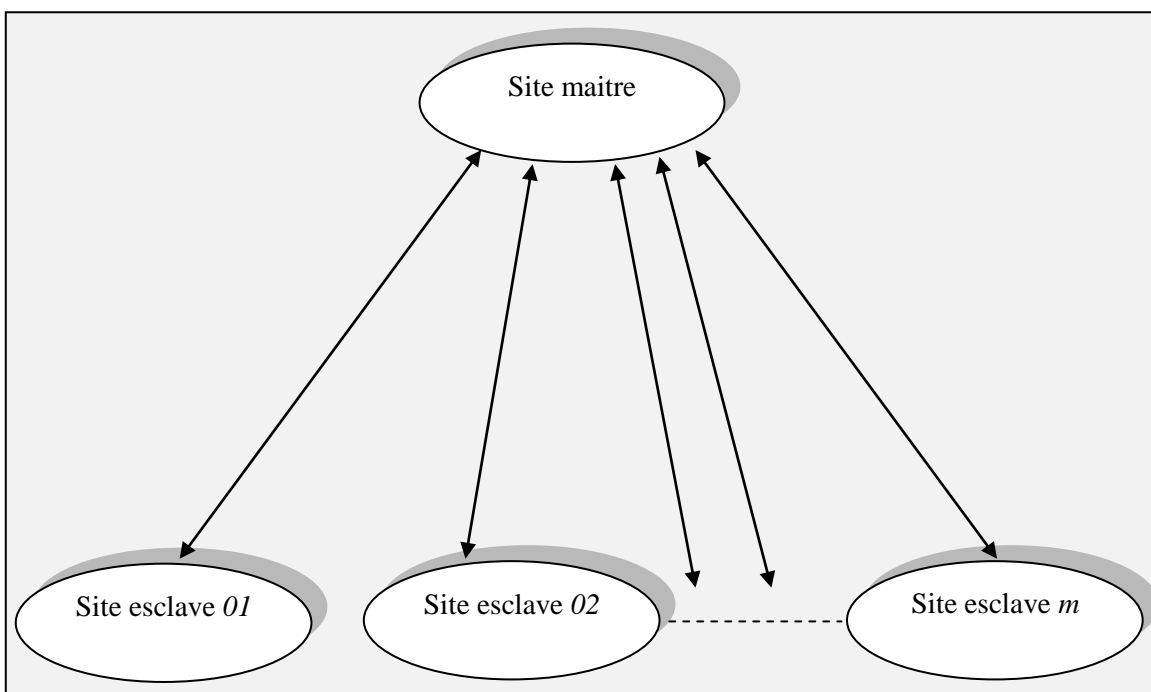


Figure 4.3. Modèle Maître / Esclave

3.2. La mise en œuvre d'un ACEVQ parallèle

Pour mettre en œuvre la version parallèle de notre algorithme proposé, on doit d'abord suivre une méthodologie de conception des algorithmes parallèles. Ainsi, nous avons appliqué la méthodologie de Foster en raison de sa popularité et de sa clarté.

Cette méthodologie consiste essentiellement en quatre étapes distinctes :

- Le partitionnement du calcul / données en tâches.
- La détermination des points de communication entre elles.
- Le regroupement des tâches qui ont une intense communication avec d'autres.
- Et enfin faire une correspondance entre ces groupes et les processeurs du système.

Nous allons examiner ces étapes en détail tout en mettant en évidence la structure de notre algorithme parallèle.

- **Partitionnement** : Le but de cette étape est de diviser le calcul en un ensemble de tâches disjointes. Dans le cas des ACEVQ, on peut intuitivement déduire n tâches, n étant la taille de la population des chromosomes quantiques. Pour chacune, on effectue une évaluation puis une interférence. La structure de chaque tâche T_i est donnée par la figure 4.4.

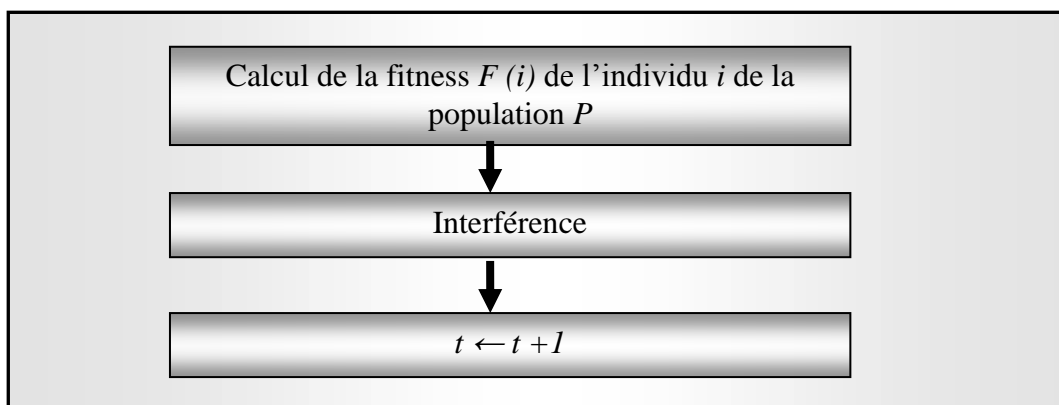


Figure 4.4. Structure des tâches parallèles

- **Communication** : On peut facilement observer que les tâches générées dans l'étape de partitionnement sont exécutables en parallèle mais elles ne peuvent pas être indépendamment exécutées : le traitement d'une tâche nécessite des données associées à d'autres tâches. On doit donc transférer des données entre les différentes tâches.

Concernant notre algorithme, la communication peut se faire en s'appuyant sur un petit protocole de communication entre les sites esclaves et le site maître, basé sur deux primitives : **Envoyer** et **Recevoir**.

Chaque site esclave accomplissant l'opération d'évaluation doit envoyer sa meilleure fitness au site maître en effectuant un **envoi** du message (**'f ; id; valeur de fitness'**) puis il se bloque en attendant la réponse du site maître, 'f' signifie ici qu'il s'agit d'un envoi d'une valeur de fitness tandis que 'id' indique l'adresse du hôte envoyant le message. Le site maître continue à recevoir les fitness des sites esclaves jusqu'à ce qu'il ne reste plus de fitness à envoyer. A ce moment là, le site maître réagit en vérifiant la meilleure fitness arrivée '**meilleure fitness**' avec celle qui existe déjà '**ancienne fitness**'. Deux cas peuvent se figurer :

- '**ancienne fitness**' \geq '**meilleure fitness**' : Dans ce cas le site maître envoie un message à tous les sites esclaves pour garder leurs meilleures solutions telles qu'elles sont (car il n'y a eu aucune amélioration), en signalant par conséquent qu'il est possible d'effectuer directement une interférence. Le format du message est tout simplement : (**'c'**).
- '**ancienne fitness**' $<$ '**meilleure fitness**' : Dans ce cas le site maître envoie un message au site esclave source de cette meilleure fitness. Son format est de (**'d'**), indiquant que le maître lui a demandé de lui renvoyer sa meilleure solution trouvée afin de la diffuser aux autres sites esclaves. Le site esclave concerné doit répondre en envoyant un message ayant le format : (**'m; id ; meilleure solution'**), 'm' indique qu'il s'agit d'un envoi d'une meilleure solution envoyée au site maître. Ce dernier réagit en envoyant cette meilleure solution arrivée à tous les autres sites esclaves permettant d'effectuer l'opération d'interférence. Le format de ce message est : (**'m; meilleure solution'**).

- **Agglomération** : Dans la phase de partitionnement, les efforts ont été focalisés sur la définition de plusieurs tâches possibles sans tenir compte du nombre de processeurs ni du coût de communication. L'étape de l'agglomération a pour but de réduire le nombre de tâches en regroupant celles ayant généralement un coût de communication élevé afin d'augmenter les performances. A la fin de cette étape on doit avoir exactement une tâche par processeur.

On doit donc reconstruire de nouvelles tâches à exécuter en regroupant un certain nombre de celles qui ont été construites dans la phase de partitionnement. La structure de chaque nouvelle tâche deviendra comme indiquée par la figure 4.5.

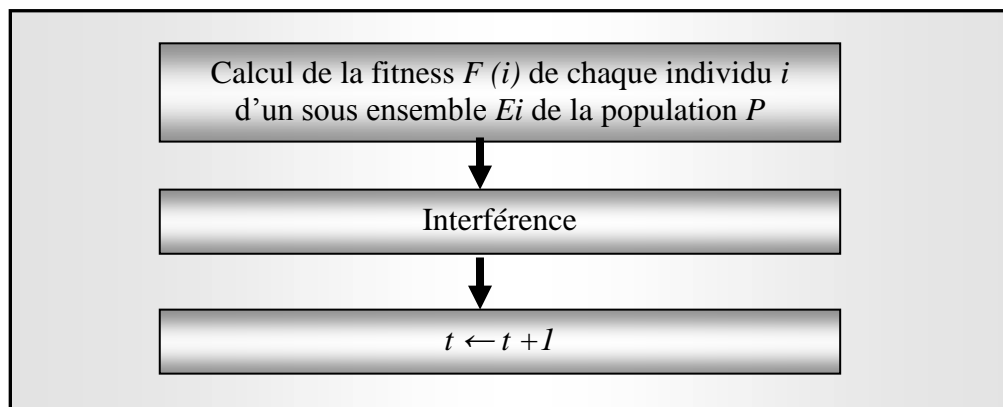


Figure 4.5. Nouvelle structure des tâches parallèles

D'où $U E_i = P$ avec $i = 1..m$, m est le nombre de processeurs. Les cardinaux des sous ensembles E_i doivent être approximativement égaux pour ne pas influencer les performances globales. On doit également tenir compte du temps écoulé par la communication entre les différents sites esclaves avec le site maître afin de mieux déduire le nombre des tâches à reconstruire.

Dans notre cas, nous avons remarqué que la taille de chaque message est relativement petite (ne dépasse pas 150 octets), et comme les câbles réseaux offrent actuellement de très grande vitesse (leur vitesse varie de $10 Mo / s$ jusqu'à $1 Go / s$), le temps de transfert des messages ne pose en principe aucun obstacle pour la performance d'exécution. Pour cela nous avons créé autant de tâches que de sites.

- **Correspondance** : Dans cette étape on doit spécifier où chaque tâche doit être exécutée. Son objectif est de minimiser le temps total d'exécution. On utilise généralement deux stratégies :

1. Placer les tâches pouvant s'exécuter en concurrence sur différents processeurs.
2. Placer les tâches qui communiquent fréquemment sur un même processeur.

Comme les différentes tâches de notre algorithme ne communiquent qu'avec le site maitre, on placera tout simplement chacune d'entre elles sur l'un des sites du réseau local d'ordinateurs.

3.3. Implémentation de notre algorithme parallèle

L'implémentation de l'algorithme parallèle proposé a été faite en Java en utilisant les *sockets* qui assurent l'ouverture des connexions entre des sites reliés par un réseau tout en permettant l'échange des messages entre eux. L'implémentation du site maitre est basée sur les *Threads* (processus légers) afin de pouvoir traiter plusieurs messages en concurrence. Le point clé de l'utilisation des mini processus est leur performance d'exécution en concurrence du fait qu'ils partagent un même espace d'adresse en mémoire centrale. Ceci implique que les changements de contextes sont très réduits et donc ne consomment pas beaucoup de temps (seul le compteur ordinal sera changé lors d'une commutation de contexte pour basculer d'une tâche vers une autre).

3.3.1. Implémentation du site maitre

Le site maitre est principalement constitué d'un ensemble de processus légers : **un dispatcher et un ensemble de traitants**. Le rôle du dispatcher est la création des traitants. A chaque site esclave tentant d'ouvrir une connexion avec le site maitre, le dispatcher crée un traitant, permettant de commencer la communication.

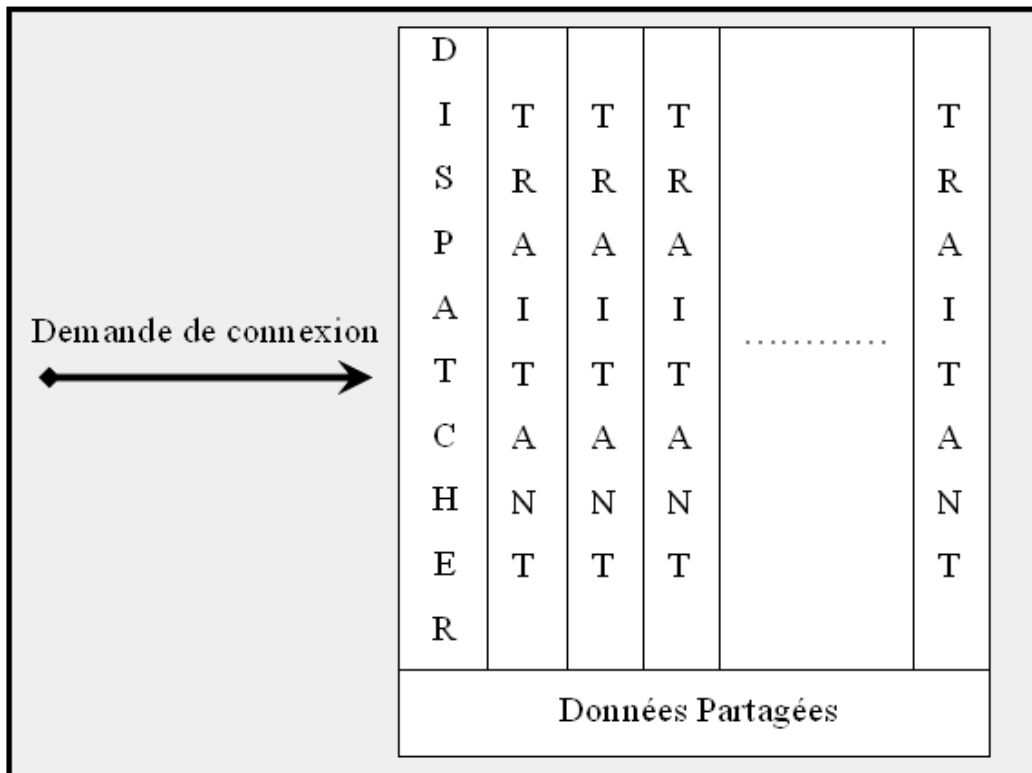


Figure 4.6. Structure du site Maitre

- **Pseudo code du site maitre**

```

Variables partagées :

Meilleure_fitness : réel ;

Meilleure_règle : chaine de caractères ;

Réponse_est_envoyée [n] : tableau de booléen ;

Réponse_est_arrivée [n] : tableau de booléen ;

Index_du_Meilleur_site : entier ;

Meilleure_règle_est_arrivée : booléen ;

Arrêt1, Arrêt2, Arrêt3 : Moniteur /* Structure de haut niveau pour la
synchronisation des processus partageant des ressources */

Initialisation des variables :

Meilleure_fitness = 0 ;

Meilleure_règle = " " ;

```

<pre> Meilleure_règle_est_arrivée = faux ; Index_du_Meilleur_site = -1 ; Pour j = 0 jusqu'à n - 1 faire Réponse_est_envoyée [j] = faux ; Réponse_est_arrivée [j] = faux ; Fin pour </pre>
<p>Programme Principal</p> <p>Début processus Maitre</p> <p>Lancer l'exécution du processus Dispatcher ;</p> <p>Fin processus Maitre</p>

Table 4.1. Pseudo code du site maitre

- **Processus Dispatcher**

<p>Début processus Dispatcher</p> <p>Pour i = 0 jusqu'à n - 1 faire</p> <p>Créer un traitant (Thread) pour le site i ;</p> <p>Fin pour</p> <p>Fin processus Dispatcher</p>
--

Table 4.2. Pseudo code du processus Dispatcher

- **Structure de chaque processus traitant i (Thread)**

<p>Début processus Traitant_i</p> <p>Tant que (! condition d'arrêt)</p>
<p><i>/* Partie I : Réception de la fitness du site esclave i */</i></p>
<p>String Réponse = Recevoir ();</p> <p>Si (début_response == "f") alors</p>

Extraire la fitness 'v' venant du site i qui a envoyé ce message

MAJ (i, v);

Fin si

Si (! Toutes_les_fitness_sont_arrivées ()) alors

Arrêt1.wait () ;

Sinon

Arrêt1. signalerTous () ; / signaler tous les processus bloqués */*

Fin si

/ Partie II : Envoi de la meilleure règle au site esclave i */*

Si (Index_du_Meilleure_site == -1) alors

Envoyer ('c') ;

Sinon

Si (Index_du_Meilleur_site != i) alors

Si (! Meilleure_fitness_est_arrivée()) alors

Arrêt2.wait () ;

Fin si

Envoyer ('b ; Meilleure_fitness ; Meilleure_règle') ;

Sinon

Envoyer ('d') ;

Réponse = Recevoir () ;

Extraire la meilleure règle venant du site i ;

Envoyer ('c') ;

Activer_arrivée_meilleure_règle () ;

Arrêt2. signalerTous () ;

Fin si

Fin si

Initialise (i) ;

<pre> /*Attente des processus qui n'ont pas encore envoyé leurs meilleures règles*/ Si (! Toutes_les_réponses_sont_envoyées ()) alors Arrêt3.wait () ; Sinon Arrêt3. signalerTous () ; Fin si </pre>
<pre> Fin tant que Fin processus Traitant_i </pre>

Table 4.3. Pseudo code de chaque processus traitant *i*

- **Ensemble des fonctions utilisées**

```

Procédure MAJ (id : entier, valeur : réel)
Début
    Si (  $\forall i$  Réponse_est_arrivée[i] = faux) alors
        Index_du_Meilleur_site = -1;
        Meilleure_règle_est_arrivée = faux ;
    Fin si
    Réponse_est_arrivée [id] = vrai;
    Réponse_est_envoyée [id] = faux;
    Si (valeur > Meilleure_fitness) alors
        Meilleure_fitness = valeur;
        Index_du_Meilleur_site = id;
    Fin si
Fin

```

```

Procédure Initialiser (id : entier)
Début
    Réponse_est_envoyée [i] = vrai ;
    Réponse_est_arrivée [id] = faux ;
Fin

```

```

Fonction Toutes_les_fitness_sont_arrivées ():booléen
Début
    Pour i = 0 jusqu'à n - 1 faire
        Si (! Réponse_est_arrivée [i]) alors
            Retourner (faux) ;
        Fin si
    Fin pour
    Retourner (vrai) ;
Fin

```

```

Fonction Toutes_les_réponses_sont_envoyées ():booléen
Début
    Pour i = 0 jusqu'à n - 1 faire
        Si (! Réponse_est_envoyée [i]) alors
            Retourner (faux) ;
        Fin si
    Fin pour
    Retourner (vrai) ;
Fin

```

```

Fonction Meilleure_fitness_est_arrivée ():booléen
Début
    Retourner (Meilleure_règle_est_arrivée) ;
Fin

```

```

Procédure Activer_arrivée_meilleure_règle ()
Début
    Meilleure_règle_est_arrivée = vrai ;
Fin

```

Table 4.4. Ensemble de fonctions utilisées par chaque processus traitant i

Le pseudo code ci-dessus montre l'existence d'un certain nombre de *variables partagées* entre les processus légers, cela nous ramène très rapidement vers le problème de la section critique où on doit gérer des accès concurrents à des variables partagées. En effet, notre algorithme est conçu de façon à ce que les trois propriétés relatives au problème de la section critique (l'exclusion mutuelle, le progrès et l'attente limitée) sont préservées.

- **Exclusion mutuelle** : L'exécution de chacune des routines déclarées est assurée d'être réalisée en **exclusion mutuelle** grâce au mot clé Java '*synchronized*' qui permet une exécution atomique d'une routine, c'est-à-dire de façon indécomposable, ce qui signifie **qu'un seul processus à la fois** peut accéder et modifier des variables partagées via la routine appelée. Les variables partagées ne sont donc jamais accédées en concurrence. Ceci signifie que la propriété d'exclusion mutuelle est totalement assurée. Il ne reste qu'à assurer les deux autres propriétés (afin de garantir l'absence de l'inter-blocage et de la famine).
- **Le progrès** : Cette propriété suppose que l'entrée d'un processus dans une section critique se fait parmi les processus demandeurs uniquement **au bout d'un temps fini**. Ceci implique que le problème d'inter-blocage devra être exclu. Etudions maintenant la structure de chaque processus afin de vérifier si cette propriété est satisfaite.

Chaque processus traitant i peut être décomposé en deux parties successives : réception de la meilleure fitness du site esclave i [partie I de la table 4.3] et l'envoi de la meilleure règle obtenue au site esclave i [partie II de la table 4.3]. Dans chaque partie, les processus sont synchronisés pour commencer leurs exécutions en même temps.

Dans le cas où l'un d'entre eux termine l'exécution de sa section critique avant les autres il sera bloqué par un appel à la primitive '*wait*' d'un des moniteurs ce qui assure **une attente passive**. Le dernier processus débloquent tous les processus bloqués derrière ce moniteur grâce à la primitive '*signalerTous*'. Cette dernière ne fait rien si elle est invoquée dans le cas où la liste des processus bloqués est vide. Nous pouvons donc conclure qu'il n'est pas possible de tomber dans une situation d'inter-blocage.

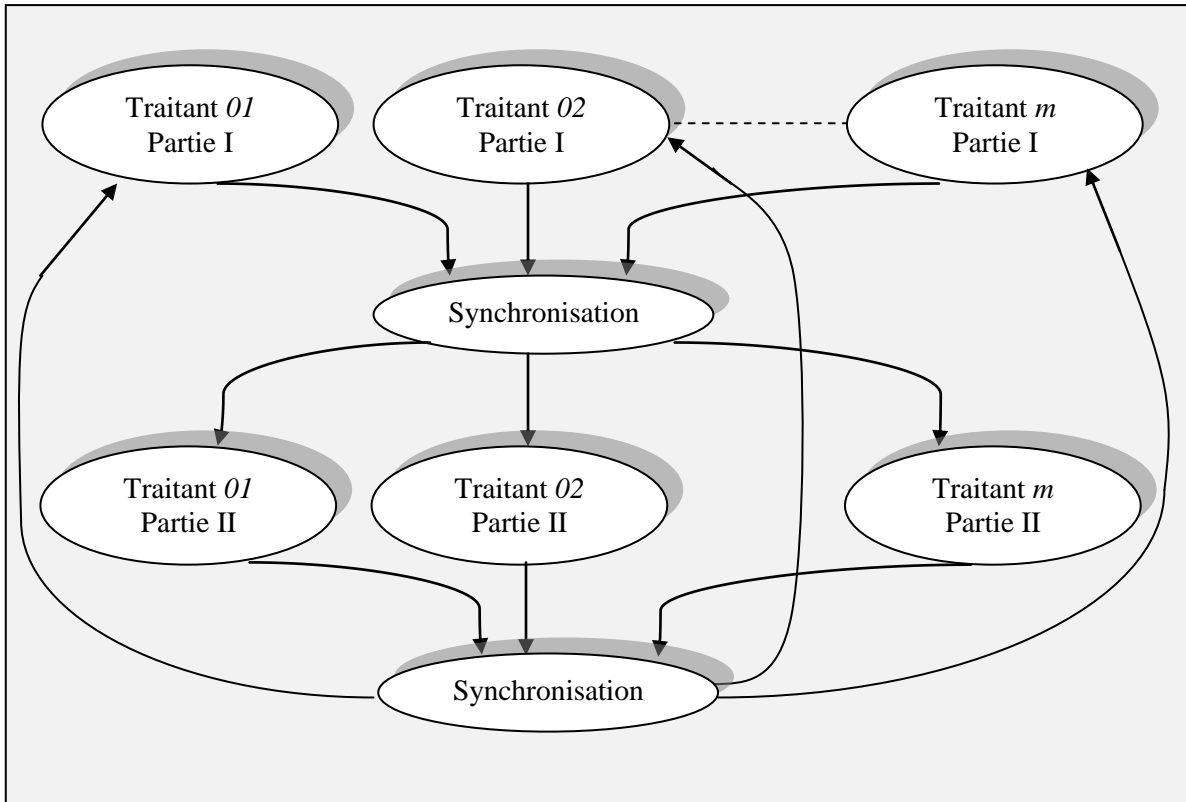


Figure 4.7. Déroulement d'exécution des processus traitants

- Attente limitée :** Cette propriété suppose qu'il y a une limite dans le nombre de fois où un processus est demandeur de permission d'entrer dans sa section critique **et que sa demande ne soit pas satisfaite**. Ceci implique que le problème de la famine devra être exclu. Notre solution vérifie cette propriété du fait qu'aucun des processus ne peut commencer l'exécution de l'une des deux parties du processus sans que les autres n'aient pas encore terminé l'exécution de l'autre partie. De plus chaque processus ne fait **qu'un seul appel au plus** à chacune des routines définies par partie. La solution proposée garantit donc l'absence de la famine entre les différents processus.

L'exécution concurrente des processus traitants peut donc s'effectuer en toute sécurité.

3.3.2. Implémentation des sites esclaves

Présentons maintenant le pseudo code de chaque site esclave i :

<p>Variables :</p> <p><i>Meilleure_fitness</i> : réel ;</p> <p><i>Meilleure_règle</i> : chaîne de caractères ;</p> <p><i>Population_de_règles</i> : structure de données qui contient les chromosomes quantiques ainsi que les configurations à évoluer.</p> <p><i>Meilleure_règle_est_arrivée</i> : Booléen ;</p> <p>Début processus Esclave</p> <p>Tant que (! condition d'arrêt)</p> <p> <i>Evoluer_population</i> () ;</p> <p> <i>Envoyer</i> ('f ; <i>Meilleure_fitness</i> ; id') ;</p> <p> <i>Meilleure_règle_est_arrivée</i> = faux ;</p> <p> Tant que (! <i>Meilleure_règle_est_arrivée</i>)</p> <p> <i>String réponse=recevoir</i> () ;</p> <p> Si (début_réponse="d") alors</p> <p> <i>Envoyer</i> ('b ; <i>Meilleure_règle</i>') ;</p> <p> Sinon</p> <p> Si (début_réponse="b") alors</p> <p> <i>Meilleure_fitness</i> = <i>Extraire la valeur</i> (v) ;</p> <p> <i>Meilleure_règle</i> = <i>Extraire la meilleure règle</i> ;</p> <p> <i>Meilleure_règle_est_arrivée</i> = vrai ;</p> <p> Sinon</p> <p> Si (début_réponse="c") alors</p> <p> <i>Meilleure_règle_est_arrivée</i> = vrai ;</p> <p> Fin si</p> <p> Fin Si</p> <p> Fin si</p> <p> Fin tant que</p> <p> <i>Interférence</i> () ;</p> <p>Fin tant que</p> <p>Fin processus Esclave</p>

Table 4.5. Pseudo code de chaque site esclave i

4. Application des ACEVQ sur divers problèmes

Dans cette section nous allons présenter, comme on l'a signalé dans le chapitre précédent, des expérimentations visant à mesurer l'efficacité des AGQ pour évoluer des AC afin de résoudre quelques problèmes calculatoires. Nous avons choisi d'étudier deux problèmes différents : le premier est le problème classique de la classification de densité alors que le deuxième est celui des bitmaps.

Pour chacun d'entre eux, nous avons évolué des populations de chromosomes quantiques en appliquant l'algorithme vu en (cf. III.7). **Le but principal est de s'assurer de la viabilité de l'algorithme proposé.**

4.1. Problème de la classification de densité

4.1.1. Les AC étudiés

Pour ce problème, nous examinons les AC dont :

- Les cellules ont deux états possibles : 0 ou 1.
- Les AC évoluent des configurations à 1-D.
- Les règles de transitions ne prennent en compte que les trois voisins directs à gauche d'une cellule et ses trois voisins directs à droite (celle-ci sera incluse), le rayon du voisinage est donc de trois ($r = 3$).

Le voisinage V_0 des cellules est donc constitué de sept cellules au total. Chaque cellule peut avoir 128 configurations différentes pour V_0 . La règle de transition doit associer à chaque configuration de V_0 la valeur que doit prendre la cellule concernée dans la prochaine génération. Par conséquent, selon la formule 2.2 (cf. II.3.2) l'espace de tous les AC possibles devra comprendre 2^{128} automates différents.

4.1.2. Codage des fonctions de transitions

Nous avons codé de tels automates sur des registres quantiques de 128 qubits où chaque qubit est associé à l'une des 128 configurations possibles de V_0 . La figure 4.8 illustre la structure de chaque chromosome quantique appartenant à la population.

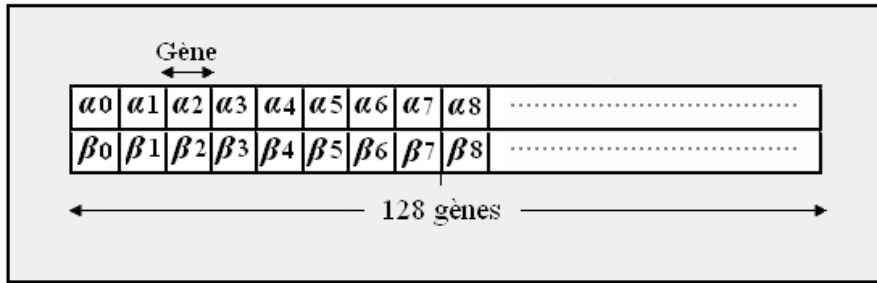


Figure 4.8. Structure d'un chromosome quantique [128 qubits]

La mesure d'un chromosome quantique conduit à créer un chromosome ordinaire dont la taille est de 128 bits.

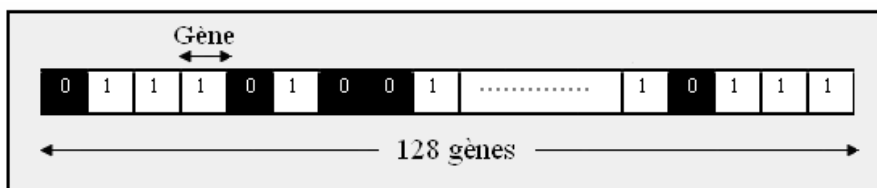


Figure 4.9. Structure du chromosome quantique mesuré résultant [128 bits]

4.1.3. Initialisation de la population

Les tests pour chaque règle de transition ont été réalisés sur des configurations de longueur $N = 149$ en respectant les conditions des limites périodiques. La taille de la population est de 100 individus, ce qui a été en réalité la taille de la population utilisée dans le test de Mitchell [16]. La population initiale est générée de manière à ce que toutes les amplitudes des chromosomes quantiques aient été initialisées par $2^{-1/2}$. Pour chaque chromosome quantique on associe un ensemble de 300 configurations initiales générées aléatoirement mais forcées d'être uniformément réparties sur $\rho \in [0.0, 1.0]$ de façon à ce que pour la moitié $\rho < \rho_c$ et $\rho > \rho_c$ pour l'autre moitié. A chaque génération, un nouvel ensemble de 300 configurations initiales est généré.

4.1.4. Evaluation des individus

L'évaluation de chaque chromosome quantique se fait, après avoir effectué une mesure, par l'évolution du chromosome extrait sur l'ensemble de ses configurations initiales pendant M itérations. On attribue comme note à un individu la fraction des cellules pour lesquelles l'état final est correct.

4.1.5. Interférence

Une fois la population évaluée, on détermine l'AC ayant la valeur d'adaptation la plus élevée, en prenant en compte les générations précédentes. Les amplitudes des individus sont mises à jour par une rotation des portes quantiques selon la table 4.6.

$x_i \ b_i$	$f(x) > f(b)$	$\Delta\theta_i$	$s(a_i \ b_i)$			
			$a_i \ b_i > 0$	$a_i \ b_i < 0$	$a_i = 0$	$b_i = 0$
0 0	0	0.005π	-	+	\pm	\pm
0 0	1	0.005π	-	+	\pm	\pm
0 1	0	0.08π	-	+	\pm	\pm
0 1	1	0.005π	-	+	\pm	\pm
1 0	0	0.08π	+	-	\pm	\pm
1 0	1	0.005π	+	-	\pm	\pm
1 1	0	0.005π	+	-	\pm	\pm
1 1	1	0.005π	+	-	\pm	\pm

Table 4.6. Table de recherche pour la rotation des portes quantiques [problème $\rho_C=1/2$]

4.2. Problème des bitmaps

Ce problème consiste à trouver un AC bidimensionnel à deux états, tel que s'il itère depuis une configuration initiale il aboutit à un état final désiré en moins de I itérations. Nous avons ainsi construit deux ensembles de configurations :

- Un ensemble de configurations sources E qui jouent le rôle d'étalons de comparaison.
- Un autre ensemble de configurations EI à base du premier en injectant du bruit selon une probabilité de 0.3 .

La figure 4.10 montre le jeu de configurations qu'on a utilisé pour l'évolution des AC. On a choisi de simples configurations car le but ici est juste de mesurer la puissance de notre algorithme pour l'exploration d'un espace de recherche d'AC.

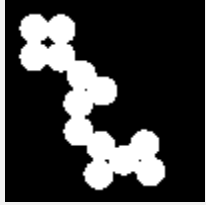


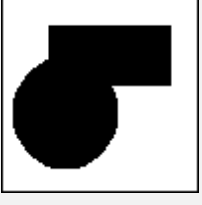




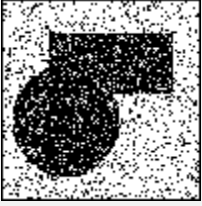

<i>E</i>					
<i>EI</i>					
	Configuration N° 1	Configuration N° 2	Configuration N° 3	Configuration N° 4	Configuration N° 5

Figure 4.10. Ensembles de configurations de taille $100 * 100$

Nous avons considéré l'ensemble *EI* comme un pool d'entraînement [40]. L'évolution sera donc faite sur toutes les configurations du pool simultanément.

4.2.1. Les AC étudiés

Pour ce problème, nous examinons les AC dont :

- Les cellules ont deux états possibles : 0 ou 1 .
- Les AC évoluent sur des grilles (configurations) à 2-D.
- Les règles de transitions ne prennent en compte que les huit voisins directs d'une cellule (celle-ci sera incluse) : c'est le voisinage de *Moore*.

Le voisinage V_o d'une cellule est donc l'ensemble des états de ses huit voisins en plus de son état, neuf cellules au total. Chaque cellule peut alors avoir 512 configurations différentes pour V_o . La règle de transition doit associer à chaque configuration de V_o la valeur que doit prendre la cellule concernée dans la prochaine génération. Par conséquent, selon la formule 2.2 (cf. II.3.2) l'espace de tous les AC possibles devra comprendre 2^{512} automates différents. Il est clair que cet espace est très large, ainsi on ajoute la contrainte suivante : les configurations de V_o seront symétriques par rapport aux rotations 90° , 180° , 270° et le retournement flip-flop horizontal et vertical, cela permet de réduire l'espace de recherche afin de mieux l'explorer.

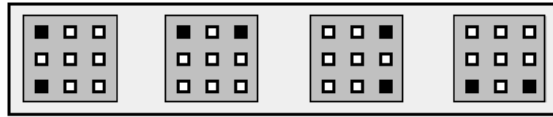


Figure 4.11. Quatre configurations du voisinage symétriquement équivalentes

Il existe 102 classes de configurations symétriques pour V_0 . Chaque classe ne sera représentée que par une et une seule configuration de V_0 parmi celles qui la constituent. Pour chaque cellule on peut donc définir 102 configurations symétriques pour V_0 , l'espace de recherche devient alors un ensemble de 2^{102} AC différents.

4.2.2. Codage des fonctions de transitions

Nous avons codé de tels automates sur des registres quantiques de 102 qubits où chaque qubit est associé à l'une des 102 classes des configurations possibles de V_0 . La figure 4.12 illustre la structure de chaque chromosome quantique appartenant à la population.

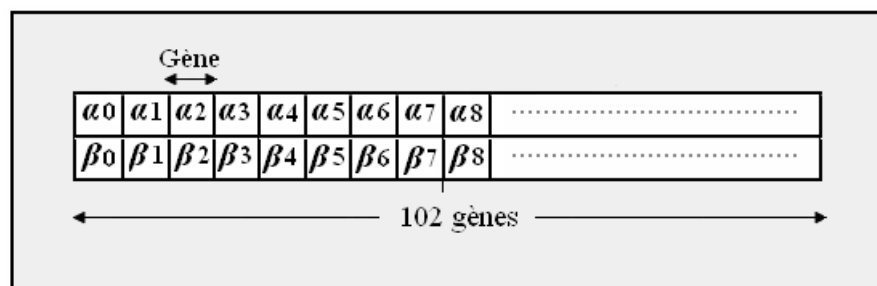


Figure 4.12. Structure d'un chromosome quantique [102 qubits]

La mesure d'un chromosome quantique conduit à créer un chromosome ordinaire dont la taille est de 102 bits.

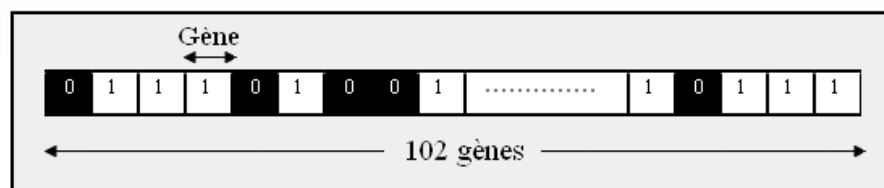


Figure 4.13. Structure du chromosome quantique mesuré résultant [102 bits]

Pour chaque configuration de V_0 on fait correspondre une position dans le chromosome pour pouvoir déterminer sa sortie, cette correspondance peut se faire par une table d'indexation où l'on associe pour chacune des 512 configurations possibles de V_0 sa position de sortie dans le chromosome (une valeur comprise entre $0 - 101$).

4.2.3. Initialisation de la population

Les tests pour chaque règle de transition ont été réalisés sur les configurations de l'ensemble EI en considérant la structure en île (cf. II.1.2). La taille de la population est de 100 individus. La population initiale est générée de manière à ce que toutes les amplitudes des chromosomes quantiques aient été initialisées par $2^{-1/2}$.

4.2.4. Evaluation des individus

L'évaluation de chaque chromosome quantique se fait en effectuant une mesure puis en évoluant le chromosome extrait sur les configurations de l'ensemble EI pendant I itérations au maximum. Les configurations résultantes seront comparées avec les configurations correspondantes (étalons de comparaison de l'ensemble E).

Notre critère de *fitness* est : $F = W + B$.

- W : Proportion des cellules blanches correctement classées dans chaque configuration de sortie.
- B : Proportion des cellules noires correctement classées dans chaque configuration de sortie.

4.2.5. Interférence

Après avoir évalué la population, on détermine le meilleur AC trouvé pour effectuer l'interférence. Les amplitudes des individus sont mises à jour selon la table 4.7.

x_i	b_i	$f(x) > f(b)$	$\Delta\theta_i$	$s(a_i b_i)$			
				$a_i b_i > 0$	$a_i b_i < 0$	$a_i = 0$	$b_i = 0$
0	0	0	0.005π	-	+	\pm	\pm
0	0	1	0.005π	-	+	\pm	\pm
0	1	0	0.08π	-	+	\pm	\pm
0	1	1	0.005π	-	+	\pm	\pm
1	0	0	0.08π	+	-	\pm	\pm
1	0	1	0.005π	+	-	\pm	\pm
1	1	0	0.005π	+	-	\pm	\pm
1	1	1	0.005π	+	-	\pm	\pm

Table 4.7. Table de recherche pour la rotation des portes quantiques [problème des bitmaps]

5. Résultats expérimentaux et discussion

5.1. Mesure d'efficacité des ACEVQ

Le but principal ici est seulement de montrer l'équivalence de notre approche avec celle des ACEV, autrement dit on veut vérifier que toute tâche pouvant être résolue par un ACEV peut aussi l'être avec un ACEVQ. Pour chacun des problèmes testés, notre algorithme a été exécuté vingt fois.

5.1.1. Problème de la classification de densité

Pour ce problème, nous avons exécuté notre algorithme pendant 100 générations. Chaque AC de la population a été itéré pendant $M = 150$ itérations. Nous avons comparé notre meilleure fitness obtenue avec celle meilleure qui existe dans la littérature. La table 4.8 montre le résultat obtenu.

Approche	Fitness
ACEVQ	94.18 %
Meilleure fitness dans la littérature (Mitchell et al 1993)	93% - 95 %

Table 4.8. Table des fitness du problème de la classification de densité

5.1.2. Problème des bitmaps

L'exécution de notre algorithme pendant 100 générations a permis d'atteindre les fitness montrées dans la table 4.9. Chaque règle dans la population a été itérée pendant un nombre maximum $I = 50$ itérations.

		C1	C2	C3	C4	C5
ACEVQ	Fitness de chaque Configuration	98.93%	98.90%	97.07%	98.98%	93.24%
	Fitness moyenne	97.424%				
ACEV	Fitness de chaque Configuration	98.98%	98.82%	96.93%	99.00%	93.28%
	Fitness moyenne	97.402%				

Table 4.9. Table des fitness du problème des bitmaps

5.2. Mesure de performance des ACEVQ (séquentiel vs parallèle)

Dans cette section nous allons exposer quelques résultats expérimentaux montrant l'utilité de la parallélisation des ACEVQ. Ainsi nous venons de prouver avec des figures illustratives que l'utilisation d'un algorithme distribué sur plusieurs sites peut atteindre un gain considérable en temps d'exécution.

La mise en marche de notre algorithme parallèle a été faite en prenant appui sur un réseau local de 10 sites : un site maître et neuf sites esclaves. La configuration matérielle et logicielle de chaque site était comme suit :

- Processeur Intel Pentium 4, 3.0 GHz.
- RAM 512 Mo.
- Système d'exploitation Windows XP SP2.
- Langage de programmation Java (JDK 1.5.0).

Nous avons considéré le problème de la classification de densité lors de la prise des mesures des temps d'exécutions. Trois tests ont été réalisés, pour chacun on varie le nombre de sites utilisés : calcul de la durée moyenne de communication entre les différents sites esclaves et le site maître, la durée moyenne d'une génération et enfin l'accélération atteignable. Avant de commencer la présentation de nos tests, on donne d'abord les tailles des populations manipulées en fonction du nombre de sites utilisés.

Nombre de sites utilisés	1	2	3	4	5	6	7	8	9
Taille de la population	100	100	99	100	100	96	98	96	99
Taille de chaque sous population	100	50	33	25	20	16	14	12	11

Table 4.10. Tailles des populations manipulées

En effet, on doit attribuer à chaque processeur la même quantité d'information à traiter afin de garantir une meilleure exploitation du parallélisme offert tout en maintenant un bon équilibre de charge.

5.2.1. Durée moyenne d'une génération

Le but de ce test est de suivre le temps total écoulé pendant une génération. Cela permet de déduire l'accélération atteignable par cette parallélisation. Nous avons considéré la moyenne des temps enregistrés dans chaque génération et pour tous les sites. Nous avons aussi calculé le temps moyen d'exécution séquentielle d'une génération pour chacune des populations construites. La table 4.11 montre les résultats obtenus :

Nombre de sites utilisés	2	3	4	5	6	7	8	9
Temps moyen d'une génération (en parallèle) (s)	304.67	200.264	151.701	122.728	96.913	84.917	73.546	66.991
Temps moyen d'une génération (en séquentiel) (s)	607.285	601.029	607.285	607.285	582.336	594.889	582.336	601.029

Table 4.11. Temps moyens d'une génération

La figure 4.14 illustre les variations du temps moyen d'une génération en fonction du nombre de processeurs utilisés. Elle comporte une exécution séquentielle et une exécution parallèle.

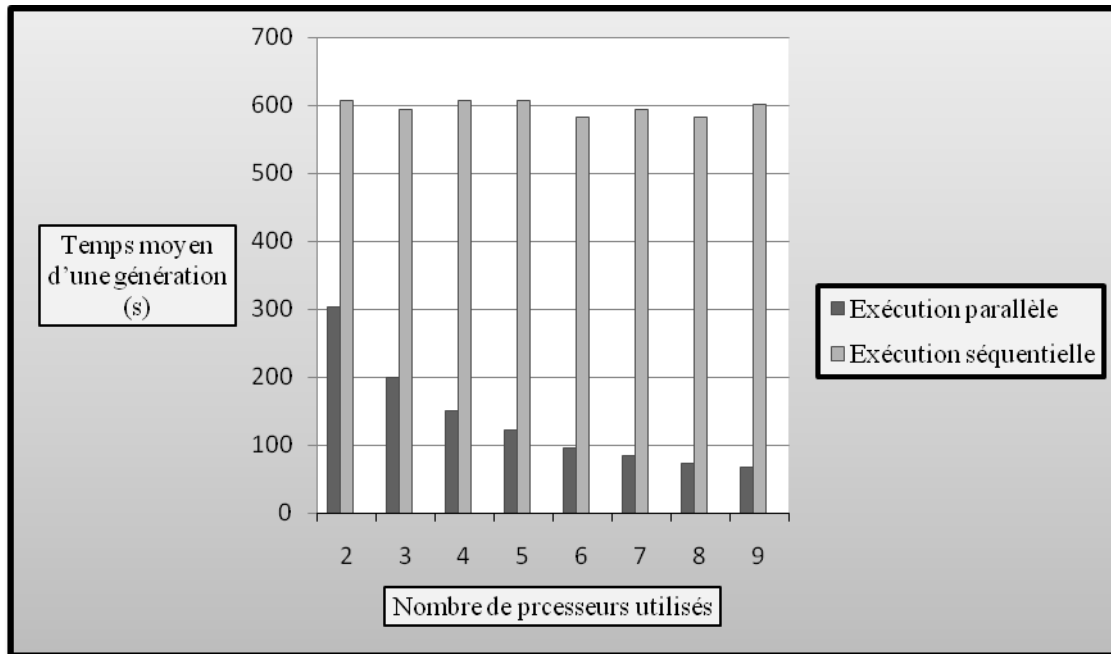


Figure 4.14. Variations du temps moyen d'une génération en fonction du nombre de sites utilisés

5.2.2. Durée moyenne de communication

Le but de ce test est de mesurer le temps écoulé par la communication entre un site esclave et le site maître en fonction du nombre des sites esclaves participants. Nous avons exécuté l'algorithme pendant 50 générations puis on a mesuré le temps moyen de communication de tous les sites et pendant toutes les générations. La table 4.12 montre les résultats obtenus :

Nombre de sites utilisés	1	2	3	4	5	6	7	8	9
Temps moyen de communication (s)	0	0.485	0.656	0.657	0.616	0.718	0.667	0.808	0.707

Table 4.12. Temps moyens de la communication

La figure 4.15 illustre les variations du temps écoulé par la communication entre les sites esclaves et le site maître en fonction du nombre de sites utilisés.

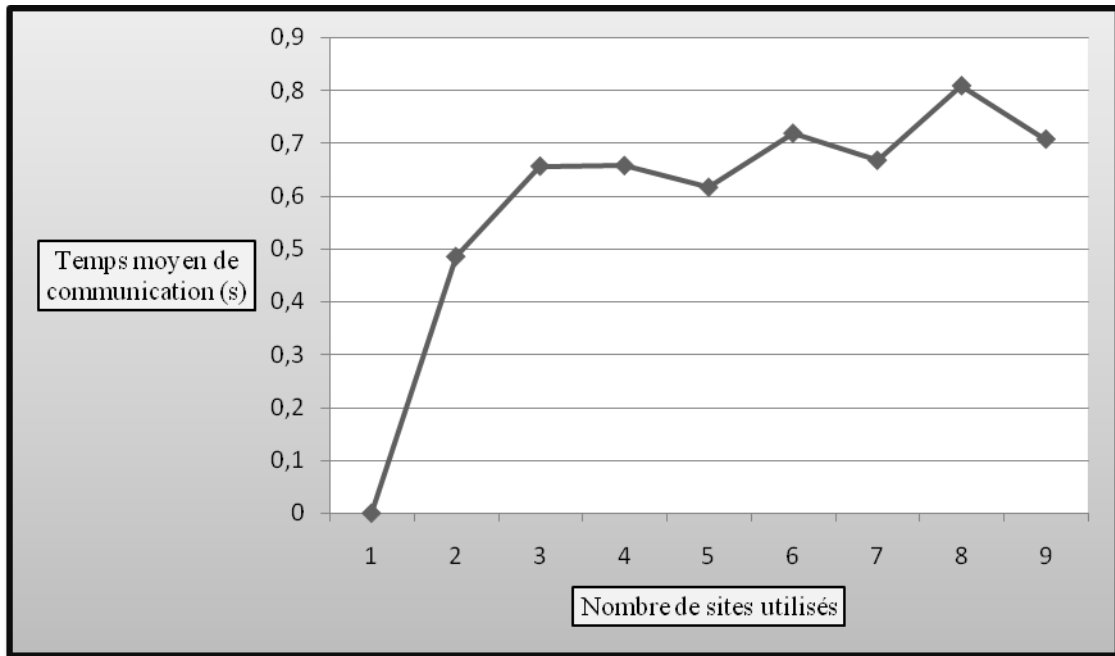


Figure 4.15. Variations du temps moyen de communication en fonction du nombre de sites utilisés

5.2.3. Accélération atteignable

Enfin, on termine par le troisième test qui consiste à calculer l'accélération atteignable. On rappelle que l'accélération est calculée par la formule 3.7 de la loi d'Amdahl vue en (cf. III.6.3.1). Pour chaque exécution, on calcule donc le rapport « *temps moyen d'une génération exécutée en séquentiel / temps moyen d'une génération exécutée en parallèle* ». La table 4.13 montre les résultats obtenus :

Nombre de sites utilisés	1	2	3	4	5	6	7	8	9
Accélération Atteignable	1	1.993	3	4.003	4.948	6.008	7.005	7.917	8.971

Table 4.13. Accélération atteignable

La figure 4.16 illustre les variations de l'accélération atteignable en fonction du nombre de processeurs utilisés.

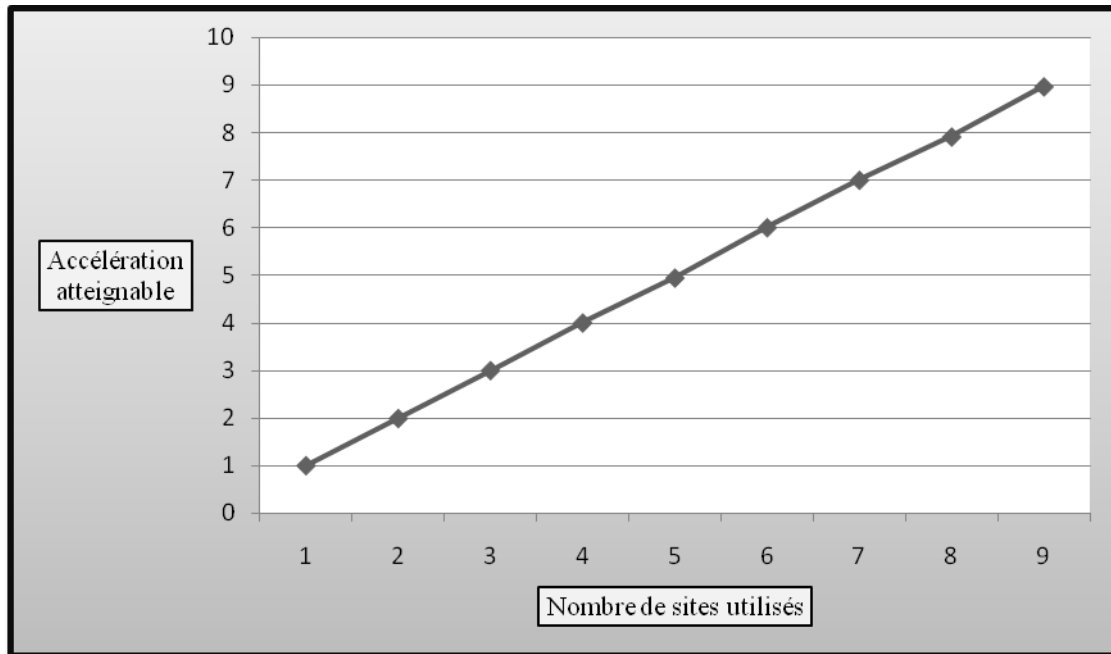


Figure 4.16. Variations de l'accélération en fonction du nombre de sites utilisés

5.3. Discussion

Concernant l'aspect efficacité, les tables 4.8 et 4.9 montrent que les fitness obtenues par des ACEVQ sont très similaires à celles obtenues par des ACEV. Le problème de la classification de densité est, comme on l'a vu, considéré comme un problème très difficile à résoudre par un AC, il peut donc être un bon critère pour mesurer la puissance de notre approche. Pour renforcer cette conclusion, un deuxième problème a été testé, celui des bitmaps. Il est clair que pour cette tâche, notre algorithme était capable de déduire les règles appropriées à sa résolution et exactement comme un ACEV. Cela prouve que les ACEVQ et les ACEV peuvent être au moins équivalents et on peut donc juger que les ACEVQ peuvent être de puissants outils pour l'exploration des espaces de recherche d'AC.

Analysons maintenant l'aspect performance, d'après la figure 4.14, on observe clairement la différence entre le temps moyen d'une génération exécutée en parallèle et le temps moyen d'une génération exécutée en séquentiel : plus le nombre de sites utilisés est augmenté, plus le temps d'exécution est diminué ce qui signifie que les performances augmentent, pour preuve, il suffit d'observer la figure 4.16 où l'accélération calculée augmente linéairement avec le nombre de sites utilisés et même tend vers le nombre de processeurs. Une seule interprétation possible à ces résultats : le temps écoulé par la phase de communication est très faible (de l'ordre de milli secondes). En effet, c'est bien ce qu'on observe dans la figure 4.15 qui montre que le temps moyen écoulé par la communication augmente de façon très lente avec

l'augmentation du nombre de processeurs utilisés. En comparant le temps moyen de communication avec le temps moyen d'une génération (exécution parallèle), on remarque que le temps moyen de communication \ll le temps moyen d'une génération, cela justifie bien nos études théoriques qu'on a mises en exergue dans le chapitre précédent (cf. III.6.3.3, la loi de Gustavson) et donc garantit une exécution hautement performante.

On pense donc que notre approche parallèle est très performante notamment si la population serait de très grande taille, comme le suggère Gustavson, ce qui permet de forcer l'accélération de tendre vers le nombre de processeurs.

On peut légitimement se demander sur l'intérêt pour les ACEVQ par rapport au modèle classique d'ACEV. Cela revient à extraire les points forts de notre approche qui n'existent pas dans le modèle classique des ACEV. Nous avons signalé dans le chapitre précédent (cf. III.3.2.2) que les algorithmes quantiques peuvent minimiser la complexité des algorithmes équivalents qui s'exécutent sur des ordinateurs ordinaires. Etudions la complexité des ACEVQ et celle des ACEV pour pouvoir estimer jusqu'à quelle niveau de réduction en termes de complexité peut-on aller.

Commençons par les ACEVQ, comme l'interférence est réalisée en se basant seulement sur la détermination du meilleur individu dans la population, cela est très similaire à la recherche d'un optimum dans un tableau ce qui signifie que la complexité d'un ACEVQ est de l'ordre de $O(N)$, N étant la taille de la population.

Pour un AG ordinaire, la complexité est calculée par l'addition des complexités des opérations de la sélection, le croisement et la mutation. Supposons que le taux de sélection des individus est de 50%, les complexités sont donc de l'ordre de $O(N^2 / 2)$, $O(N / 2)$ et $O(N / 2)$ pour les trois opérations respectivement. La complexité totale de l'algorithme est donc de l'ordre de $O(N*(N / 2 + 1))$.

En comparant les deux complexités, il est évident que la complexité de notre algorithme est de $O(N)$ tandis que celle d'un AG est de $O(N^2)$, et on pense donc que ce résultat est très intéressant car **la complexité ici a été réduite pour devenir linéaire.**

On signale ici que l'exécution des deux approches (ACEVQ et ACEV) de manière parallèle (cas du modèle maître / esclave) n'a aucune influence sur la valeur de la complexité, car pour

les ACEVQ la détermination du meilleur individu n'est possible qu'après l'envoi des toutes les fitness de la part des sites esclaves, tandis que pour un ACEV les opérations de sélection, croisement et mutation sont entièrement réalisées au niveau du site maître (les esclaves ne font qu'évaluer les individus). Cela veut dire que notre algorithme possède toujours une complexité moindre que celle d'un ACEV.

6. Conclusion

En conclusion, les ACEVQ peuvent constituer un très puissant outil pour l'exploration des espaces de recherche d'AC. Nous avons vu que leur structure offre une possibilité d'exploiter efficacement le parallélisme existant tout en préservant une haute efficacité. Nos expérimentations ont suggéré qu'ils peuvent dépasser les difficultés induites par l'exécution parallèle des ACEV. On remarque aussi leur simplicité de mise en œuvre séquentielle et même parallèle du fait que la plupart des opérations génétiques ont été omises (ni sélection, ni croisement ni mutation), de plus l'architecture de leur version parallèle est relativement simple (modèle maître / esclave) sachant que l'exécution parallèle n'est plus un obstacle pour nous.

Conclusion générale

Ce mémoire nous a permis de retracer l'évolution des ACEV, depuis leur invention à la fin des années 80 jusqu'aux développements plus récents. Tout d'abord conçus pour répondre à un problème bien spécifique, celui de la classification de densité, les ACEV ont conduit à lancer un grand concours qui a débuté en 1993, en coïncidence avec la première grande publication par Mitchell et ses collègues à SFI, pour la découverte de nouvelles règles montrant l'existence d'un calcul émergent. Ils ont ensuite lâché la bride à leurs potentiels, et leur valeur comme outil puissant pour l'exploration des espaces de recherches d'AC s'est fait ressentir dans un grand nombre de domaines scientifiques allant des sciences de la complexité à l'informatique.

En effet, les AC ont pour autant été utilisés pour étudier des systèmes complexes, modéliser des phénomènes et même résoudre des problèmes calculatoires. Il semble que seule l'utilisation de l'ordinateur comme outil fondamental peut vaincre cette variété de problèmes. Nous pensons donc que les ACEV sont un domaine de recherche d'avenir, qui n'a livré jusqu'à présent que quelques secrets et il reste alors encore beaucoup de choses à faire.

Le travail présenté dans ce mémoire a eu pour objectif de proposer une nouvelle approche basée sur les AGQ pour la conception des AC. En effet, le calcul quantique a évacué une grande partie des préoccupations des chercheurs en raison de sa puissance pour minimiser la complexité des algorithmes s'exécutant sur des machines classiques. A notre tour, nous avons utilisé ce concept afin d'en introduire un autre : **les automates cellulaires évolutionnaires quantiques (ACEVQ)**.

Comme nous l'avons vu, les ACEVQ sont basés sur la représentation des AC par des chromosomes quantiques puis effectuer une interférence sur chacun des individus de la population afin de former la génération suivante. Notre but principal est de profiter de la mise en œuvre parallèle des ACEVQ et baptisée **des automates cellulaires évolutionnaires quantiques parallèle (ACEVQP)**. Dans la pratique, l'exécution des programmes parallèles n'est plus devenue un obstacle en raison de la vigueur des machines modernes notamment les réseaux d'ordinateurs et les grilles de calcul.

Afin de mesurer la pertinence de notre approche une série d'expérimentations a été faite sur deux problèmes différents : le problème classique de la classification de densité et le problème des bitmaps. Nos résultats expérimentaux ont montré une nette amélioration par rapport au modèle classique, notamment lors d'une exécution distribuée, traduite par :

- Une simplicité de la mise en œuvre (absence des opérations de la sélection, le croisement et la mutation).
- Une bonne accélération (une meilleure exploitation du parallélisme).
- Possibilité d'exécution sur des ordinateurs quantiques.

Comme perspective à ce travail nous pensons à tester la version parallèle des ACEVQ sur des grilles de calculs afin de vérifier la satisfaction de la propriété de **la scalabilité**, car ici les tests ont été intentionnellement choisis simples (en raison de l'absence des outils nécessaires pour effectuer des tests sur des grilles de calculs), le but étant seulement de démontrer la faisabilité d'une telle parallélisation.

Une autre perspective concerne l'exécution parallèle des ACEVQ sur des réseaux d'ordinateurs ayant différentes topologies afin de pouvoir déduire la topologie la plus appropriée notamment pour la phase d'envoi des messages.

Nous espérons enfin que notre application pourra être utile à la communauté étudiant les AC, ne serait-ce que comme point de départ, pour la réalisation d'une véritable application dans notre laboratoire qui en a bien besoin.

*« Ce n'est pas la fin. Ce n'est même pas le commencement de la fin.
Mais, c'est peut-être la fin du commencement »*

Winston Churchill

Bibliographie

- [1] Charles Darwin, "On The Origin of Species", 1859.
- [2] A.M Turing, "On Computable Numbers, with an application to the Entscheidungsproblem", *Proceedings of the Mathematical Society*, Série 2, Vol. 42, p.230-265, 1936.
- [3] Fogel, L., Owens, A.J., Walsh, M.J., "Artificial Intelligence through Simulated Evolution", Wiley, 1966.
- [4] M. Gardner, "Mathematical Games: The fantastic combinations of John Conway's new solitaire game life", *Scientific American*, pp. 120-123, Octobre 1970.
- [5] Rechenberg, I. "Evolutions strategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution". Fromman- Holzboog Verlag, Stuttgart, 1973.
- [6] Holland, John H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.
- [7] P. Gacs, G. L. Kurdyumov, and L. A. Levin. "One-dimensional uniform arrays that wash out finite islands". *Probl. Peredachi. Inform.*, 14:92–98, 1978.
- [8] S. Wolfram. "Universality and complexity in cellular automata". In *Physica D*, 10:1–35, 1984.
- [9] Wolfram S., Packard N. H., "Two-Dimensional Cellular Automata", *Journal of Statistical Physics*, vol. 38, p. 901-946, 1985.
- [10] Wolfram S., "Twenty Problems in the Theory of Cellular Automata", *Physica Scripta*, vol.9, p. 170-183, 1985.
- [11] Albert J. and Culik K. "A simple universal cellular automaton and its one-way and totalistic version". *Complex Systems*. Vol. no.1: 1_16, 1987.
- [12] K. Culik, S. Yu, "Undecidability of CA classification schemes", *Complex Systems*, 2, p.177, 1988.
- [13] N. H. Packard. "Adaptation towards the Edge of Chaos". In J. A. S. Kelso, A. J. Mandell, and M. F. Shlesinger, editors, *Dynamic Patterns in Complex Systems*, pages 293-301. World Scientific, Singapore, 1988.
- [14] M. Mitchell, J. P. Crutchfield, and P. T. Hraber. "Dynamics, computation, and the edge of chaos : A re-examination". In G. Cowan, D. Pines, and D. Melzner, editors, *Integrative Themes*, Reading, MA, Addison-Wesley. In press, 1993.

- [15] S.Rasmussen, C. Knudsen, and R. Feldberg. "Dynamics of programmable matter". In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 211-254, Redwood City, CA. Addison-Wesley, 1992.
- [16] Melanie Mitchell, Peter T. Hraber, and James P. Crutchfield, "Revisiting the Edge of Chaos : Evolving Cellular Automata to Perform Computations", *Complex Systems*, 7, 89--130, 1993.
- [17] J. P. Crutchfield and J. E. Hanson, "Turbulent pattern bases for cellular automata". *Physica D*, in press, Santa Fe Institute Report SFI-93-03-010, 1993.
- [18] Sahota, P, Daemi, M.F, Elliman, D.G, "Training Genetically Evolving Cellular Automata for Image Processing", 1994 International Symposium on Speech, Image Processing and Neural Networks, Page(s):753 - 756 vol.2, 1994.
- [19] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. "Evolving globally synchronized cellular automata". In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336-343, San Francisco, CA, 1995.
- [20] Land, M. and Belew, R.: "NO Two-State CA for Density Classification Exists". *Physical Review Letters* 74, 5148, 1995.
- [21] M. Mitchell, J. P. Crutchfield, and R. Das, "Evolving cellular automata to perform computations: A review of recent work," in *Proc. 1st Int. Conf. Evol. Computat. Appl. (EvCA'96)*. Moscow, Russia: Russian Academy of Sciences, 1996.
- [22] M. Sipper. Co-evolving Non-Uniform Cellular Automata to Perform Computations. *Physica D*, 92:193-208, 1996.
- [23] Mathieu S. Capcarrere, Moshe Sipper, and Marco Tomassini. Two-state, $r=1$ cellular automaton that classifies density. *Phys. Rev. Lett.*, 77(24) : 4969 – 4971, December 1996.
- [24] M.Sipper, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, Springer-Verlag, Heidelberg, 1997.
- [25] H. Fuks, "Solution of the Density Classification Problem with Two Cellular Automata Rules". *Physical Review E*, 55(3), 2081-2084, 1997.
- [26] J. Paredis, "Coevolving cellular automata: Be aware of the red queen," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Bäck, Ed. San Mateo, CA: Morgan Kaufmann, pp. 393–400, 1997.

- [27] G. VAN De VIJVER - *Emergence et explication* - Intellectica : Emergence and explanation, 1997/2 n°25, ISSN n°0984-0028 185-194, 1997.
- [28] H. Juillé and J. B. Pollack, “Coevolving the ‘ideal’ trainer: Application to the discovery of cellular automata rules,” in Proc. 3rd Annu. Conf. Genetic Programming, J. R.Koza,W. Banzhaf, K. Chellapilla, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. L. Riolo, Eds. San Mateo, CA: Morgan Kaufmann, 1998.
- [29] Wei WU, “Synthèse d’un contrôleur flou par Algorithme Génétique : Application au réglage dynamique des paramètres d’un système”, thèse de doctorat, université de Lille1, France, 1998.
- [30] Werfel. J, Mitchell.M and Crutchfield. J. P. “Resource sharing and coevolution in evolving cellular automata”. Submitted to IEEE Trans. Evol. Comp, 1999.
- [31] K.H. Han, Genetic Quantum Algorithm and Its Application to Combinatorial Optimization Problem, IEEE Proc. Of the 2000 Congress on Evolutionary Computation, pp. 1354-1360, 2000.
- [32] Fates Nazim, “Les automates cellulaires vers une nouvelle épistémologie ?”, Mémoire de DEA, université de Paris I Sorbone, France, 2001.
- [33] Niloy Ganguly, Biplab K Sikdar, Andreas Deutsch, Geo_rey Canright, P Pal Chaudhuri, “A Survey on Cellular Automata,” Centre for High Performance Computing, Dresden University of Technology, Tech. Rep. 9, 2003.
- [34] C.L.Chang; Y.J.Zhang; Y.Y.Gdong, “Cellular automata for edge detection of images”, 2004 International Conference on Machine Learning and Cybernetics, Page(s): 3830 - 3834 vol.6, 2004.
- [35] E. Sapin, O. Bailleux, J.J. Chabrier, and P. Collet. “A new universal cellular automata discovered by evolutionary algorithms”. Gecco2004. Lecture Notes in Computer Science, 3102:175–187, 2004.
- [36] E. Sapin, “Approche évolutionniste de la recherché d’automates cellulaires universels”. Revue Technique et Sciences Informatiques (TSI). Volume X – n° X/2005, pages 1 à 28, 2005.
- [37] Thomas Bäck, Ron Breukelaar and Lars Willmes, Inverse Design of Cellular “Automata by Genetic Algorithms: an Unconventional Programming Paradigm”, International Workshop UPP 2004, Revised Selected and Invited Papers, Jean-Pierre Banâtre et al. (editors), Springer-Verlag GmbH LNCS 3566, pg. 161—172, 2005.
- [38] Paul L Rosin, “Training Cellular Automata for Image Processing”, IEEE Transactions on Image Processing, VOL. 15, NO. 7, Page(s): 2076 – 2087, 2006.

- [39] R.J.Chen; Y.H.Chen; C.S.Chen; J.L.Lai, "Image Encryption/Decryption System Using 2-D Cellular Automata", IEEE Tenth International Symposium on Consumer Electronics, Page(s):1 – 6, 2006.
- [40] M. Batouche, S. Meshoul and A. Abbassene, "On solving edge detection by emergence", the 19th International Conference on Industrial Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE'06), Annecy, (France), 27-30, Lecture Notes in Artificial Intelligence (LNAI 4031), pp.800-808, 2006.
- [41] Abdesslem Layeb and Djamel-Eddine Saidouni, "Quantum Genetic Algorithm for Binary Decision Diagram Ordering Problem". IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.9, 2007.

Webographie

- [42] <http://www.vieartificielle.com/article/?id=88>.
- [43] www.wikipedia.com.
- [44] http://www.futura-sciences.com/fr/definition/t/high-tech-1/d/loi-de-moore_2447/.
- [45] http://gameoflife.free.fr/pres_fr.htm.
- [46] http://www.cinum.org/fr/fiche_3_informatique_quantique/7-10-41.html
- [47] <http://chezmatthieu.blogspot.com/2007/02/info-ou-intox-le-premier-ordinateur.html>.
- [48] rolab.free.fr/Project/Licence/Quantum/Quantique.ppt.
- [49] <http://www.rennard.org/alife>.
- [50] http://taxules.free.fr/cours_MP/tipe/algogene.html.
- [51] <http://www.cornu.eu.org/news/le-jeu-de-la-vie-et-l-emergence-de-la-complexite>.
- [52] <http://www.greyc.ensicaen.fr/>.
- [53] <http://magnin.plil.net/spip.php?article45>.
- [54] <http://www.futura-sciences.com/comprendre/d/dossier285-1.php>
- [55] <http://automatesintelligent.blog.lemonde.fr/2007/03/05/un-ordinateur-quantique-commercialise-des-2008/>.
- [56] <http://www.presence-pc.com/actualite/D-Wave-Orion-21814/>.

Annexes

25. Annexe A

26. Annexe B

Annexe A : Quelques stratégies calculatoires obtenues par l'étude du problème $\rho c = \frac{1}{2}$

Voici quelques stratégies calculatoires découvertes par l'équipe « The Evolving Cellular Automata Group » à l'institut SFI lors de l'étude du problème de la classification de densité.

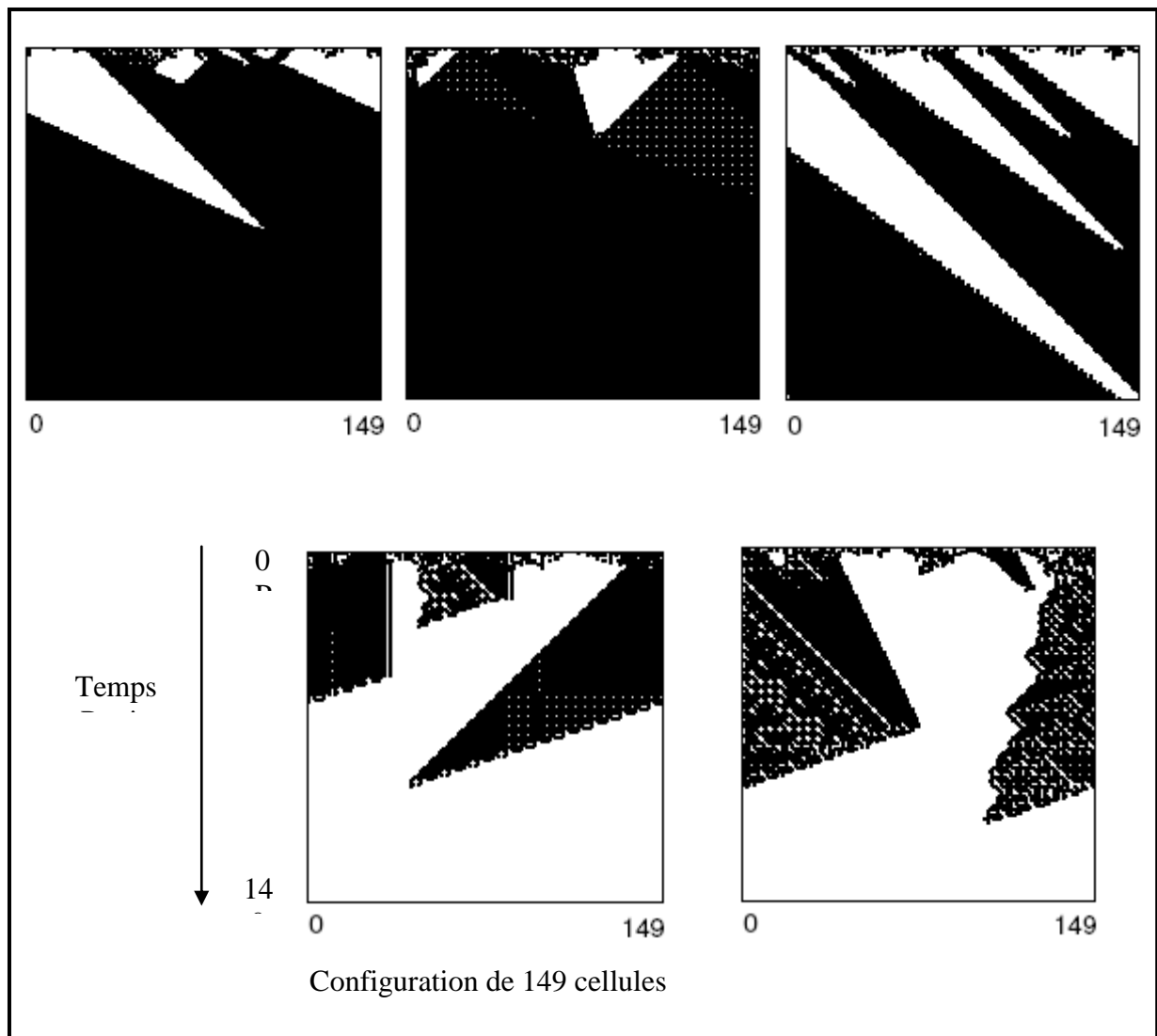
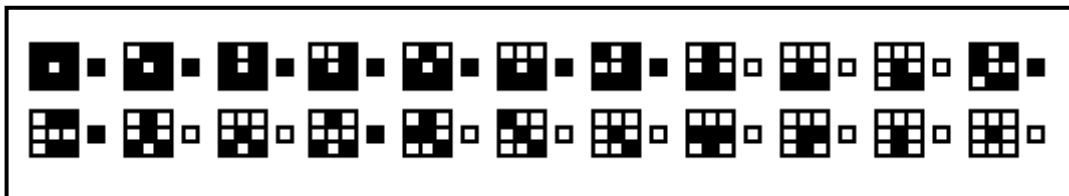


Figure A.1. Stratégies calculatoires découvertes à partir de l'étude du problème $\rho c = \frac{1}{2}$

Annexe B : Viabilité de l'AC optimal retenu par l'ACEVQ

L'application des ACEVQ sur le problème des bitmaps a permis de retenir l'AC illustré par la figure B.1.



La figure B.2 illustre l'ensemble des configurations résultantes.

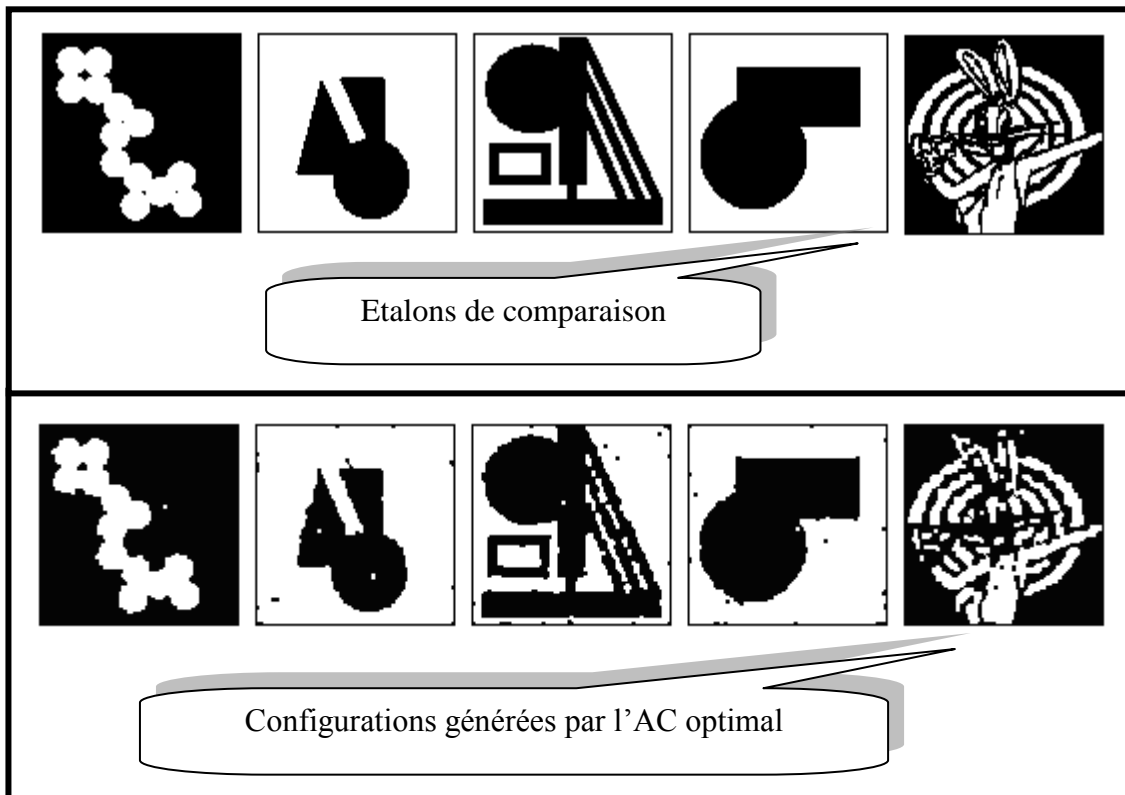


Figure B.2. Configurations résultantes du problème des bitmaps par l'AC optimal

عرفت دراسة الأنظمة المعقدة اهتماما كبيرا من طرف الباحثين في مجال علوم التعقيد و كذلك الإعلام الآلي. تعتمد دراسة مثل هذه الأنظمة على تمثيلها عن طريق نماذج و من ثم استخراج الخصائص المرغوب في دراستها. تعتبر الآلات الخلوية أداة قيمة لدراسة مثل هذه الأنظمة لأنها يمكن أن تسفر عن سلوك جد معقد انطلاقا من مجموعة قواعد يمكن أن تكون بسيطة جدا. ومع ذلك ، فإن صعوبة استغلال إمكاناتها على نحو فعال غالبا ما حد من استخدامها في معالجة العديد من الإشكاليات. لذلك فكر العلماء في إيجاد طريقة آلية لتصميمها عن طريق الخوارزميات التطورية و التي بإمكانها استكشاف عالم الآلات الخلوية و نتكلم إذا عن الآلات الخلوية التطورية. مهمتنا من خلال هذا البحث هي أساسا القيام بوضع نهج يتسم بفعالية وكفاءة لتصميم الآلات الخلوية. النهج المقترح في هذا البحث (الآلات الخلوية التطورية الكمية) يعتمد أساسا على استعمال الخوارزميات الجينية الكمية من أجل تصميم آلات خلوية و هذا في نظام يعمل بمبدأ التوازي. النتائج التجريبية أظهرت تحسنا ملحوظا مقارنة بالنموذج الكلاسيكي المستعمل في تصميم الآلات الخلوية من خلال أداء عالي التوازي مع الحفاظ على الكفاءة.

كلمات مفتاحية : نظام معقد، آلة خلوية ، الخوارزميات التطورية ، إعلام آلي كمي.

Résumé

L'étude des systèmes complexes a récemment connu un intérêt croissant par des chercheurs en sciences de la complexité et de l'informatique. Leur étude passe généralement par leur modélisation. Il semble que la classe du modèle abstrait d'automates cellulaires constitue un outil précieux pour l'étude de tels systèmes dans la mesure où ils peuvent exhiber un comportement global énorme partant seulement d'un ensemble de règles pouvant être très simples. Cependant, la difficulté d'exploiter efficacement leur potentiel a souvent limité leur utilisation. Pour surmonter ce problème, les chercheurs ont pensé d'automatiser leur processus de conception en faisant appel aux algorithmes évolutionnaires qui peuvent explorer des espaces de recherche des automates cellulaires et on parle alors des automates cellulaires évolutionnaires.

Notre travail consiste essentiellement à développer une approche qui soit à la fois efficace et performante pour la conception des automates cellulaires. L'approche proposée (automate cellulaire évolutionnaire quantique) repose sur l'évolution des automates cellulaires par des algorithmes génétiques quantiques dans un environnement parallèle. Les résultats expérimentaux ont montré une nette amélioration par rapport au modèle classique des automates cellulaires évolutionnaires dans la mesure où ils s'exécutent dans des environnements parallèles en haute performance tout en préservant l'aspect efficacité.

Mots-clés : Système Complexe, Automate Cellulaire, Algorithme Evolutionnaire, Informatique Quantique.

Abstract

The study of complex systems has recently known a growing interest by researchers in sciences of complexity and information technology. Their study is usually done by modeling. It seems that the class of the abstract model of cellular automata is a valuable tool for studying such systems as they can produce a huge global behavior from only a set of rules that can be very simple. However, the difficulty of effectively exploit their potential is often limited their use. To overcome this problem, researchers have thought to automate their design process using evolutionary algorithms that can explore research spaces of cellular automata; this technique is called evolutionary cellular automata.

Our research work is basically to develop an approach that is both effective and efficient for designing cellular automata. The proposed approach (quantum evolutionary cellular automata) is based on evolution of cellular automata by quantum genetic algorithms in a parallel environment. The experimental results have shown a marked improvement over the classical model of evolutionary cellular automata as they can be run in parallel environments with high performance while preserving their efficiency.

Keywords: Complex System, Cellular Automata, Evolutionary Algorithm, Quantum Computing.