République Algérienne Démocratique et Populaire

Ministère de L'Enseignement Supérieur et de La Recherche Scientifique

Université Mentouri Constantine

Faculté des Sciences de l'Ingénieur

Département : Informatique

N° d'ordre : **145/mag/2006**

N° de série : **011/inf/2006**

Mémoire pour l'obtention du diplôme de Magister en Informatique

Option : Information et Computation

Présenté par :

LABED SAID

Sujet du mémoire :

Systèmes Complexes Adaptatifs Application au traitement des images

Dirigé par : **Pr Mohamed Batouche**

Soutenu le : 04/05/2006 devant le jury composé de :

- Pr **Mahmoud Boufaida**	Professeur	Université Mentouri Constantine	Président
- Pr **Mohamed Batouche**	Professeur	Université Mentouri Constantine	Rapporteur
- Dr **Djamel Eddine Saidouni**	Maître de conférence	Université Mentouri Constantine	Examinateur
- Dr **Salim Chikhi**	Maître de conférence	Université Mentouri Constantine	Examinateur

Résumé

Le cadre général de ce mémoire de Magister est l'intelligence artificielle distribuée (I.A.D). Nous nous intéressons au développement des systèmes de calcul distribués, aptes à fonctionner dans un environnement complexe et dynamique, et à faire face aux imprévus, ceci dans l'absence d'un agent central de contrôle et de coordination. Dans ce contexte nous avons introduit une approche ascendante "Calcul orienté autonomie" A.O.C comme paradigme de modélisation et de résolution des problèmes difficiles. A.O.C emploie la notion d'autonomie comme noyau de modélisation de comportement dans n'importe quel système complexe, ainsi les interactions locales entres les entités autonomes représentent la force de cette approches. Dans ce mémoire nous avons exploité l'approche A.O.C pour réaliser une tâche qui consiste à segmenter une image à niveau de gris, en s'inspirant du phénomène de diffusion dans la nature. L'image est représentée par un tableau à deux dimensions dont chaque élément est un pixel. Une population d'entités autonomes est déployée sur l'image tel que chaque entité est capable d'estimer si un pixel appartient ou non à une région homogène en interagissant avec son environnement représenté par son voisinage. Lorsqu'une région non homogène est rencontrée, les entités autonomes diffusent vers un autre pixel selon certaine direction, dans le but de localiser une région homogène. Lorsque une entité autonome rencontre une région homogène elle va reproduire un certain nombre de progénitures qui seront diffusés selon une certaine direction, et le pixel sur lequel elle réside sera étiqueté.

Mots clés : Autonomie, Calcul orienté autonomie, Comportement, Diffuser, Entité autonome Environnement, Reproduire, Segmentation d'images, Système complexe

---

Abstract

This work takes place in the field of Distributed Artificial Intelligence. We have interested in development of distributed computational systems witch able to work in complex and dynamical environment in the absence of a central controlling agent. We introduce autonomy oriented computing (AOC) as a paradigm to describe systems for solving hard computational problems and for characterizing the behavior of a complex system. AOC employs autonomy as the core model of any complex system, and local interactions between the autonomous entities are the primary driving force of AOC. Formulation of an autonomy oriented computing system involves an appropriate analogy with nature. In This work, an autonomy oriented approach has been used to tackle the task of image segmentation. A digital image is viewed as a two-dimensional cellular environment in witch autonomous entities inhabit and attempt to label homogenous segments. Autonomous entities operate directly on the individual pixels of digital image by continuously sensing their neighboring regions and checking the homogeneity criteria of relative contrast, regional mean, and regional standard deviation. When a non-homogeneous region is found, the autonomous entity will diffuse to another pixel in a certain direction within the local region. In contrast, when an entity locates a homogeneous region, it replicates (breeds) itself to give rise to certain number of offspring and deliver them to its local region in a certain direction.

Key words :Autonomy, Autonomous entity, Autonomy Oriented Computing, Behavior, Breed, Complex system, Diffuse, Environment, Image segmentation,

Remerciements

*Je voudrais, avant toute chose, exprimer ma plus grande reconnaissance à **Mr Mohamed Batouche** pour avoir accepté de diriger ce mémoire avec efficacité et rigueur. Merci également pour son soutien et ses conseils.*

*Je tiens aussi à remercier **Mr Mahmoud Boufaida** pour avoir accepté de présider mon jury, dont j'apprécie tout particulièrement la rigueur scientifique et les grandes qualités humaines. Je n'oublierai pas le soutien amical et chaleureux qu'il m'a toujours témoigné.*

*Je tiens également à exprimer ma gratitude à **Mr Djamel Eddine Saidouni**, et qui m'a fait le plaisir de participer à mon jury, et qui m'a fait l'honneur d'évaluer ce travail.*

*Je voudrais aussi remercier **Mr Salim Chikhi** d'avoir accepté de m'accorder une partie de son temps précieux pour être membre de mon jury, et qui m'a fait l'honneur d'évaluer ce travail. Merci également pour ses commentaires et ses conseils.*

Je tiens aussi à remercier toute ma famille pour l'affection et la patience qu'elle m'a accordé depuis le début de mes études, et en particulier mes parents, qui m'ont toujours fait confiance, ma femme et mes enfants : Yasser et Norhane

Table des matières

Liste des figures

Liste des tables

.

INTRODUCTION GENERALE

Introduction générale

Actuellement les logiciels informatiques sont réalisés pour des applications de plus en plus complexes, distribuées, ayant une très forte dynamique et manipulent une grande quantité d'information, caractérisées par leur aspect coopératif, adaptatif et leur ouverture. Dans le domaine de la robotique, on s'intéresse à la réalisation des systèmes permettant à un ensemble de robots de coordonner leurs comportements dans le but d'achever une tâche ou une mission globale d'une façon cohérente. Dans le domaine de l'Internet, on s'intéresse à la réalisation des applications complexes telles que le commerce électronique, la recherche sur le Web et l'optimisation des ressources distribuées. Dans la médecine, le défi est de réaliser des applications de monitoring permettant une surveillance efficace et rigoureuse des patients. Enfin dans le domaine des réseaux on s'intéresse actuellement au déploiement des réseaux mobiles et sans fil.

Toutes ces applications ont des caractéristiques communes, à savoir la tâche de computation est répartie, et qu'il n'existe pas une seule machine dédiée pour accomplir cette tâche.

Le succès de ces applications est lié au déploiement à grande échelle des agents de calcul capables d'une manière autonome de prendre des décisions locales pour atteindre le but global.

D'où la nécessité de concevoir des logiciels aptes à fonctionner dans un environnement à très forte dynamique, à faire face à des imprévus et adaptés pour résoudre des problèmes complexes, distribués pour lesquels un contrôle global est impossible à mettre en œuvre.

Les systèmes multi-agents classiques ont apporté une nouvelle manière de concevoir de tels logiciels. En effet, le comportement du système est le résultat des interactions et du comportement des agents qui le constituent. Pour le concepteur, la tâche est simplifiée par le fait qu'il doit uniquement développer les agents et les mécanismes d'interaction entre ces derniers

Actuellement, il est de plus en plus difficile - sinon impossible - de contrôler correctement l'activité de ces logiciels situés dans des environnements de plus en plus dynamiques. Pour faire face à ces difficultés, une solution consiste à laisser plus d'autonomie

aux logiciels afin qu'ils s'adaptent au mieux aux imprévus et donc concevoir des systèmes multi-agents adaptatifs. En effet, les systèmes multi-agents classiques prennent bien en compte la complexité des interactions, mais la dynamique n'est que très peu présente.

Donc, Il s'agit de développer des systèmes capables de travailler "en intelligence" avec leur environnement : les agents interagissent dans un environnement incertain et dynamique, le système doit être capable de s'auto-observer afin de détecter les phénomènes émergents et adapter son comportement et sa structure à ces phénomènes et à l'évolution de son environnement.

D'où la nécessité de développer un nouveau paradigme dans le but d'inventer de nouvelles technologies, de découvrir de nouvelles lois scientifiques et de bien comprendre notre univers. Ce nouveau paradigme est appelé A.O.C « Autonomy Oriented Computing », introduit par Jiming Liu en 2001. A.O.C est une approche ascendante pour l'étude des systèmes complexes, l'objectif est de développer des algorithmes ou systèmes de calcul permettant la modélisation et la résolution des systèmes complexes et distribués à grande échelle ceci en considérant l'autonomie comme le noyau de modélisation de comportement dans n'importe quel système complexe, et en s'inspirant des phénomène d'autonomie et d'auto-oganisation dans la nature [15].

A.O.C permet de :

- Comprendre le sens d'autonomie dans les systèmes naturels et artificiels
- Résoudre les problèmes difficiles et modéliser les systèmes complexes.
- Développer des modèles de calcul autonome.
- Etudier les caractéristiques d'un phénomène complexe ou d'un comportement émergent dans les systèmes naturels et artificiels qui nécessitent l'auto-organisation et un grand nombre d'interactions entre entités.
- Découvrir les lois et les mécanismes derrières un phénomène complexe ou un comportement émergent.

En d'autres termes, A.O.C permet de modéliser, expliquer et prédire les comportements dans un système complexe adaptatif

Dans notre problématique, nous nous intéressons aux agents réactifs, adaptatifs et situés dans un environnement dynamique et réel, la communication entre eux est très simple et

disposant les mêmes capacités de perception, ceci dans l'absence d'un contrôleur central. Notre tâche est de faire apprendre aux agents des comportements spécifiques pour agir sur l'environnement en exécutant des règles de comportement, dont le but essentiel est d'aboutir à la structure émergente voulue sous forme de comportement cohérent et complexe. Il s'agit donc de développer des systèmes informatiques adaptatifs, des systèmes capables d'adapter automatiquement leur comportement en fonction de la dynamique réelle de leurs interactions avec l'environnement

Quand ce type d'adaptation opère pendant que le système agit dans son environnement, on dit que le système utilise son expérience pour réaliser un apprentissage en ligne. Cet apprentissage lui permet de conserver un comportement adapté à l'accomplissement de la tâche qui lui est dévolue, même en cas de pannes ou de changements imprévus dans l'environnement. L'apprentissage est nécessairement incrémental, c'est-à-dire que le système doit apprendre au fur et à mesure de son expérience, sans pouvoir compter sur une base d'exemples pertinents préalablement établis.

Notre travail est organisé à travers ce mémoire comme suit :
Dans le chapitre 1, nous avons détaillé les différentes notions nécessaires à savoir : la notion de système complexe, système complexe adaptatif, agent, systèmes multi-agent, systèmes multi-agent adaptatifs et enfin la notion d'apprentissage et de rétroaction.

Le chapitre 2 est consacré pour les méthodes d'apprentissage des agents réactifs et homogènes.

Dans le 3ème chapitre, nous avons détaillé le nouveau paradigme A.O.C « Autonomy Oriented Computing » [14].

Enfin dans le 4ème chapitre, nous avons exposé notre problème complexe à résoudre qui consiste à localiser les différentes régions homogènes dans une image à niveau de gris, avec mise en œuvre de la solution proposée en se basant sur le nouveau paradigme A.O.C.

Chapitre I

SYSTEMES COMPLEXES ADAPTATIFS

1. Systèmes complexes adaptatifs
Complex Adaptive System C.A.S

1.1 Définition d'un système complexe [1] : Un système complexe est un système composé d'un ensemble d'éléments homogènes ou hétérogènes, interagissant entre eux de façon non linéaire, mettant ainsi en œuvre une dynamique permettant à l'ensemble de système d'exister comme un tout, différent de la simple somme de ses composants. On distingue alors deux niveaux :

- un niveau micro, représentant le niveau des composants, avec des propriétés locales à chacun d'eux.
- Un niveau macro, représentant l'ensemble du système, avec des propriétés nouvelles, que l'on ne trouve dans aucun des composants pris individuellement. On parle alors d'émergence de nouvelles propriétés [36].

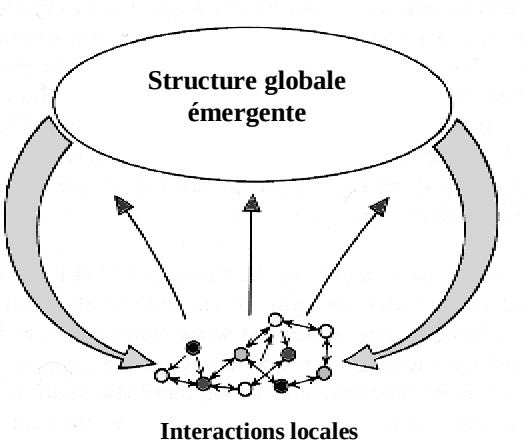


Figure 1.1 : ***Schématisation d'un système complexe****.*

Un exemple typique d'un système complexe : est la volée d'oiseaux (Figure 1.2), on constate l'absence d'un oiseau qui joue le rôle de leader et que les oiseaux suivent des règles très simple à savoir :

- La séparation : l'oiseau cherche à s'éloigner des voisins trop proches ;
- L'alignement : l'oiseau cherche à régler sa direction comme celle de la moyenne de ses voisins ;
- La cohésion : l'oiseau cherche à se rapprocher des voisins trop éloignés

D'où l'émergence d'un comportement complexe.

Figure 1.2 : ***Exemple de volée d'oiseaux***

Un autre exemple est celui d'une colonie de fourmis où chaque fourmi se comporte selon les trois règles suivantes :

1) Promener aléatoirement
2) Si nourriture trouvée, alors prendre une pièce et retourner à la colonie en déposant des traces chimiques (phéromones) dans leur environnement, qui s'évaporent avec le temps, puis aller à la règle 1
3) Si trace de phéromone trouvée alors suivez-là pour aller à la nourriture.
   Aller à la règle 2

D'où l'émergence du phénomène du plus court chemin

1.2 Agent

1.2.1 Définition d'un agent : On peut noter qu'il n'existe pas une définition universelle d'un agent. Nous présentons quelques définitions :

- "Un Agent est une entité qui perçoit son environnements et agit sur celui-ci " [3].
- "Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome pour atteindre les objectifs pour lesquels il a été conçu "[4].
- "Tout ce qui peut être vu comme percevant son environnement à l'aide de capteurs et agissant sur cet environnement a` l'aide d'effecteurs, de façon autonome"

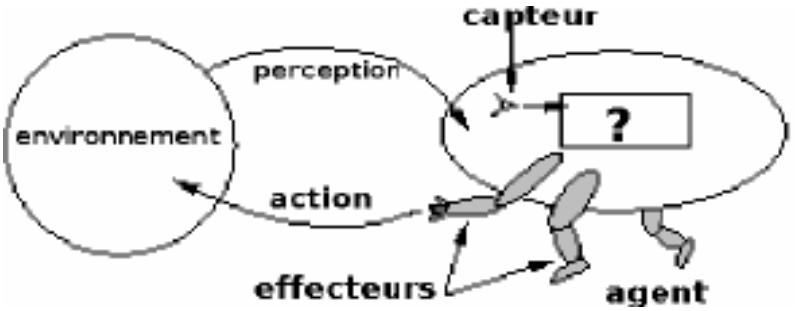


Figure 1.3 : ***Schématisation d'un agent (Russell and Norvig).***

Jacques Ferber propose une définition minimale commune [5]

Un agent est une entité physique ou virtuelle :

1- qui est capable d'agir dans un environnement;
2- qui peut communiquer directement avec d'autres agents;
3- qui possède des ressources propres
4- qui est capable de percevoir son environnement
5- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune)
6- qui possède des compétences et des offres de services
7- qui peut éventuellement se reproduire

* L'introduction de la notion d'agent est motivée par :

- la complexité et la répartition des systèmes
- la distribution des ressources
- la distribution de l'expertise
- l'interopérabilité avec les systèmes existants (Legacy systèm)
- la nécessité de personnaliser les interfaces
- La dynamique de l'environnement (Robot Cup)
- pas d'algorithmes connus pour certains problèmes (gestion d'une fourmilière)

1.2.2 Caractéristiques des agents : Un agent est [29] :

- **Situé** : capable d'agir sur son environnement à partir des entrées sensorielles qui reçoit de ce même environnement. Les agents situés possèdent une représentation souvent très limitée de leur environnement mais rarement des capacités de raisonnement sur cette représentation ; ils sont pour cela traditionnellement qualifiés de réactifs. Un agent est réactif lorsqu'il agit uniquement en fonction de stimuli (internes et externes), sans avoir besoin de représentation symbolique ou d'historique

- **Autonome** : l'agent est capable d'agir sans l'intervention externe des autres agents ou des humains, il contrôle ses actions en fonction de son état interne et de son environnement. Cette autonomie implique évidemment que l'agent doit être capable de gérer les conflits éventuels entre ses objectifs et son état interne ou l'état de son environnement. Les mécanismes permettant de sélectionner des actions et de résoudre les conflits font l'objet de nombreuses recherches. J.Ferber distingue par ailleurs cinq catégories de motivations, c'est-à-dire de raisons qui poussent un agent à agir [40]. Nous en retiendrons trois :

  - **les motivations personnelles** tendent à satisfaire les besoins et objectifs internes de l'agent;

  - **les motivations environnementales** sont produites par ce que perçoit l'agent de son environnement ;

  - **les motivations sociales** découlent de l'organisation des agents imposée par le concepteur du système.

- **Intelligent** : Un agent est habituellement qualifié d'intelligent lorsqu'il possède des capacités de représentation symbolique et de raisonnement (agent cognitif), l'agent doit exhiber un comportement proactif, capable de prendre l'initiative au bon moment pour atteindre son but (résolution orienté but)
- **Sociable** : il doit être capable d'interagir avec d'autres agents (logiciels ou humains) afin d'accomplir des tâche ou aider les autres agents à accomplir les leurs.
- **Capable de répondre à temps** : l'agent droit être capable de percevoir son environnement et d'élaborer une réponse dans le temps requis
- **Mobile** : peut se déplacer d'un environnement à l'autre
- **Rational** : apte à choisir une action en tenant compte de son état interne
- **Capable pour un comportement non déterministe**
- **Coordinateur**
- **Coopérateur**
- **Adaptatif** : Un agent adaptatif est capable de modifier son comportement et ses objectifs en fonction de ses interactions avec son environnement et avec les autres agents. En particulier, il peut réagir différemment selon l'agent avec lequel il interagit. Cette caractéristique nécessite des capacités d'apprentissage, par exemple pour élaborer des cartes cognitives ou pour modifier ses préférences comportementales en fonction de son

expérience. L'adaptation peut également être collective : en utilisant des mécanismes d'évolution, des populations d'agents peuvent apprendre à s'adapter à leur environnement.

1.2.3 Types d'agent

D'après les caractéristiques citées ci-dessus, on peut avoir différents types d'agents [30] :

- les agents cognitifs
- les agents réactifs
- les agents autonomes
- les agents mobiles
- les agents adaptatifs
- les agents flexibles
- les agents sociaux
- les agents coordinateurs
- ……

Mais généralement, les agents sont classés selon deux courants de pensée : l'école cognitive influencée par l'I.A symbolique et l'école réactive influencée par la vie artificielle et l'intelligence "sans représentation".

- **Agent cognitif** : l'agent dispose d'une capacité de raisonnement, il possède des informations et des savoir- faire nécessaires à la réalisation de sa tâche et à la gestion des interactions avec son environnement et les autres agents. L'agent possède un but à atteindre à l'aide des plans explicites.

- **Agent réactif** : c'est une composante très simple qui perçoit l'environnement et est capable d'agri sur celui- ci, il n'a pas une représentation symbolique de l'environnement et des connaissances. L'intelligence est distribuée entres plusieurs agents réactifs, autrement dit le comportement intelligent devrait émerger de l'interaction entre les agents actifs et l'environnement. Les agents réactifs ne disposent que d'un protocole de communication réduit. La relation avec l'environnement répond à la loi stimulus/ réponse.

1.2.4 Environnement

L'environnement est l'ensemble des conditions extérieures susceptibles d'agir sur le fonctionnement d'un système. Un agent autonome évolue de façon continue dans un environnement, dans lequel peuvent exister d'autres agents. Puisqu'un agent agit sur son

environnement et que l'environnement agit sur l'agent, il existe une réelle interaction entre ces deux entités. La connexion entre un agent et son environnement peut être plus ou moins forte selon la façon dont l'agent acquiert l'information en provenance de l'environnement. A partir des informations obtenues, l'agent peut créer et modifier sa propre représentation de l'environnement. La connexion agent-environnement est forte dans le cas d'un agent situé, c'est-à-dire lorsque l'agent possède des capteurs et des effecteurs. Les capteurs sont capables de percevoir directement (mais souvent partiellement) les éléments de l'environnement ou les modifications survenues dans cet environnement. Les effecteurs permettent à l'agent d'agir sur son environnement

Les principales caractéristiques de l'environnement sont [31] [35] :

- Accessible / inaccessible : l'accessibilité signifie que l'agent ait un accès à l'état complet de l'environnement
- Déterministe / non déterministe : un environnement est déterministe c'est le prochain état de l'environnement est complètement déterminé par son état courant et l'action sélectionnée par l'agent.
- Statique / dynamique : un environnement est dit dynamique s'il peut changer pendant la prise de décision de l'agent.
- Discret / continu : s'il existe un nombre limité de perceptions et d'actions possible, on parlera d'environnement discret

1.2.5 Communication

Il existe deux types de communication :

1.2.5.1 Communication directe : La communication directe correspond à l'envoi volontaire d'un message à un ou plusieurs agents. Il s'agit d'une conversation entre agents cognitifs, capables d'interpréter des actes de langage et de raisonner sur des messages ambigus. Un acte de langage est une action intentionnelle effectuée au cours d'une communication, par exemple une requête effectuée auprès d'un agent avec l'espoir d'obtenir un service. Une conversation est un échange de messages, généralement structuré selon un protocole de communication.

Les langages de communication entre agents, ou ACL (Agent Communication Languages), définissent en particulier des types de messages indiquant l'intention de l'agent émetteur (requête, ordre, information spontanée, proposition de service,affirmation, croyance, etc.). C'est le cas du langage KQML (Knowledge Query and Manipulation Language), du langage de la FIPA (Foundation of Intelligent Physical Agents).

1.2.5.2 Communication indirecte : La communication indirecte se fait par l'intermédiaire de l'environnement. Elle correspond à l'envoi d'un signal dans l'environnement ou à la modification de l'environnement (création, destruction ou modification d'objets). Un signal peut être soumis à des règles de propagation ; par exemple, son intensité peut décroître en fonction de la distance entre l'émetteur et le récepteur ou en fonction du temps (dans le cas de phéromones). Ce type de communication peut être involontaire, et ainsi participer à une coopération de type réactive.

1.3 Système multi-agents S.M.A

1.3.1 Définition d'un système multi-agents

Un système multi- agent est un système distribué composé d'un ensemble d'agents est caractérise par :

- chaque agent a des informations ou des capacités de résolution de problèmes limités
- il n'y a aucun contrôle global du système
- les données sont décentralisées
- le calcul est asynchrone

Jacques ferber propose la définition suivante [5] : un système multi- agent se compose de :

- Un environnement **E**
- Un ensemble d'objet **O**, ces objets sont situés, cela signifie que pour tout objet il est possible, à un moment donnée d'associer une position dans **E**, ces objets peuvent être perçus, crées, détruits et modifies par les agents .
- un ensemble d'agents **A** qui sont des objets particuliers représentant les objets actifs du système (**A** est un sous ensemble de **O**)
- un ensemble d'opération **Op** permettant aux agents de **A** de percevoir, produire, consommer, transformer et manipuler les objets de **O**.
- des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification que l'on appellera les lois de l'univers.

Ainsi la définition suivante est donnée par Ferber :

"Un S.M.A est un ensemble d'entités (physiques ou virtuelles) appelées agents, partageant un environnement commun (physique ou virtuel), qu'elles sont capables de percevoir et sur lequel elles peuvent agir. Les perceptions permettent aux agents d'acquérir des informations sur l'évolution de leur environnement. Les agents interagissant entre eux directement ou

indirectement, et exhibent des comportements corrélés créant ainsi une synergie permettant à l'ensemble des agents de former un collectif organisé."

1.3.2 Caractéristique des SMA :

- Interaction : sous forme de coordination, de communication et d'organisation (centralisée ou décentralisée)
- Adaptation et apprentissage

1.3.3 Domaines d'application des S.M.A

1- résolution de problèmes :

- de décision : par planification distribuée des tâches.
- d'optimisation

2- Modélisation et simulation d'entités distribuées : on part généralement d'une population d'individus quelconques (par exemple en éthologie ce sont des insectes, en sociologie ce sont des gens) que l'on modélisé dans leur structure interne et leur comportement (réactif/ cognitif), on associe à chaque individu un agent, la phase de simulation nécessite l'implémentation des loi du monde qui est simulé, c'est une phase discrète car à chaque pas de temps simulé, le système multi- agent évolue. L'objectif est d'aboutir à la phase de visualisation qui consiste à montrer les entités qui sont sensées émerger de la population sous formes de structures ou de comportements.

3 – Applications distribuées :

- Gestion de grands systèmes industriels
- Contrôle du trafic aérien.
- Applications de monitoring : surveillance médicale
- Réseaux de télécommunication
- Recherche de l'information
- Commerce électronique
- La robotique
- Traitement d'image et réalité augmentée.
- Applications Multimédia

1.4 Système complexe adaptatif (Complex Adaptive System (C.A.S)

1.4.1 Définitions

Définition 1 :

Kevin Dooley donne dans [2] une définition d'un système complexe adaptatif :

"A CAS behaves/evolves according to three key principles: order is emergent as opposed to predetermined, the system's history is irreversible, and the system's future is often unpredictable. The basic building blocks of the CAS are agents. Agents are semi-autonomous units that seek to maximize some measure of goodness, or fitness, by evolving over time. Agents scan their environment and develop schema representing interpretive and action rules. These schemas are often evolved from smaller, more basic schema. These schemas are rational bounded : they are potentially indeterminate because of incomplete and/or biased information; they are observer dependent because it is often difficult to separate a phenomenon from its context, thereby identifying contingencies; and they can be contradictory. Schema exist in multitudes and compete for survival."

Un système complexe adaptatif est un système complexe ayant les principes fondamentaux suivants :

- L'émergence de l'ordre, l'irréversibilité de l'historique du système et l'imprévisibilité de son comportement futur;
- Ces éléments constituants de base sont des agents qui cherchent à maximiser la performance du système.
- Les agents forment de schémas globaux, constituant des règles d'interprétation ou d'action, par agrégation de schémas plus primitifs.
- L'existence de multiples schémas pouvant être contradictoires, et qui sont en compétition pour se maintenir dans l'environnement ;
- La sensibilité au contexte et la dépendance de l'observateur.

Définition 2

"Un système complexe adaptatif est un système complexe ayant l'aptitude de s'adapter aux changements de l'environnement. Ces systèmes ayant comme caractéristique principale l'auto organisation et le contrôle distribué. Les agents interagissent selon les règles de comportement et s'adaptent en changeant leurs règles de comportement. L'expérience des agents est exploitée pour déterminer le comportement futur".

Les agents dans un système complexe adaptatif doivent être capables de réagir aux changement et perturbation de leur environnement qui est complexe et dynamique. L'adaptation du comportement ou de la structure interne permet ainsi d'acquérir dynamiquement les connaissances qui n'ont pas été fournies par le concepteur. Cette adaptation entraîne l'émergence de comportements globaux qui sont parfois indésirables, et cela dans l'absence d'un contrôleur centralisé.

Le système doit être en effet capable de s'auto-obsderver afin de détecter les phénomènes émergents et à l'évolution de son environnement. La modélisation de ces systèmes complexes nécessite ainsi l'adaptation au niveau local (Micro) et au niveau global (Macro) [7]

Un élément essentiel dans un système complexe est la fonction d'adéquation (fitness) de chaque agent et de tout le système. Cette fonction permet de mesurer l'adéquation des agents à chaque niveau d'observation.

* Quelques exemples des systèmes complexes adaptatifs :

- Les systèmes immunitaires
- Les colonies d'insectes
- Les marchés économiques décentralisés
- Les robots
- Le système nerveux

1.4.2 Concepts de base d'un système complexe adaptatif

Un système complexe adaptatif est basé sur les concepts suivants :

a- Agents : qui représentent des entités autonomes qui interagissent pour exécuter leurs taches

b- Adaptation : les agents sont capables d'agir dans leur environnement et de changer leur comportement en se basant sur leur processus d'apprentissage. Il existe quatre formes d'adaptation :

- par réaction direct à l'environnement. Typiquement un comportement réactif est exprimé par la formule : **Quand** événement arrivé**, si** condition(s**) alors** action. Des exemples d'agents réactifs : les thermostats, les robots et les machines à laver.
- par raisonnement : possibilité de réaliser des inférences malgré la nature réactive des agents
- par apprentissage : les agents peuvent apprendre.

- par évolution : des modifications peuvent être effectuées sur des générations successives d'agents (Algorithmes génétiques).

Ces formes d'adaptation peuvent être utilisée séparément ou combinées.

c- Emergence : Un phénomène est dit émergent si [6] :

Il y a un ensemble d'entités en interaction dont la dynamique est exprimée dans un vocabulaire ou théorie D distincte du phénomène émergent à produire ;

La dynamique de ces entités interagissantes produit un phénomène global qui peut être un état structuré stable ou même la trace d'exécution ;

Ce Phénomène global peut être observé et décrit par un vocabulaire ou théorie D' distincte de la dynamique sous-jacente.

Les stocks des marchés, les systèmes immunitaires et les colonies des fourmis sont des exemples d'agents agissant individuellement, et l'interaction entre eux engendre un nouveau phénomène.

Les propriétés de l'émergence sont :

- dans une structure émergente, les agents sont organisés tel que le tout est plus grand que la somme des parties, autrement dit les parties n'aboutissent pas séparément à une structure émergente, d'où l'importance de l'interaction entre agents.
- les règles de comportement sont très simples
- l'approche ascendante "Bottom-Up" est mieux adaptée pour concevoir des systèmes émergents
- les structures émergentes peuvent être eux-mêmes des composants pour d'autres structures.
- les structures émergentes sont constituées d'agents homogènes (les colonies de fourmis) ou hétérogènes (les systèmes immunitaires).

d- La nature : considérée comme source d'inspiration pour la résolution et la conception des systèmes complexes. Par exemple British Telecom a exploitée le modèle des fourmis et phéromone sur son réseau téléphonique, tel que les appels effectués avec succès laissent des traces pour les appels futurs.

1.5 L’apprentissage

La problématique de l'apprentissage et de l'adaptation dans les systèmes multi-agents s'est considérablement développée ces dernières années. Ce thème présente un intérêt et une importance pour les S.M.As dans lesquels l'adaptation est une caractéristique importante pour assurer l'autonomie. Dans beaucoup de travaux, l'apprentissage se fait de manière supervisée, où un observateur extérieur juge le comportement de l'agent et donne une récompense qui motive l'apprentissage. Cet observateur joue le rôle d'un professeur qui observe son élève, et juge l'efficacité des actions de celui-ci, ce qui le permet de le récompenser ou de le punir.

Dans un système apprenant autonome (non supervisé), il faut chercher les motivations qui vont permettre à une entité d'apprendre par elle-même et de développer ces connaissances. En effet on aimerait que l'intelligence émerge de motivations relativement simples, mais ayant une forte signification pour l'entité apprenante. Pour cela il faut donc que l'entité "vive" suffisamment longtemps et qu'elle rencontre beaucoup de situations qui vont la faire se développer. La réactivité de ces entités leur permet de réagir directement aux variations de l'environnement. Cet apprentissage se fait de façon non supervisée, c'est à dire qu'il n'y a pas de hiérarchie au sein du système. Les agents apprennent par leurs propres moyens, ils sont totalement autonomes, et ne nécessitent pas l'intervention d'un opérateur pendant l'apprentissage.

1.6. Phénomène de rétroaction (Feedback)

On parle de feedback lorsque le système interagit avec son environnement, les entrées sont le résultat de l’influence de l’environnement sur le système, et les sorties sont l’influence du système sur l’environnement [34].

La rétroaction peut définir comme l’action qui agit en retour sur le phénomène qui se trouve à sa source.

On peut avoir deux types de rétroaction :

A. **Rétroaction positive** : lorsque les nouvelles données agissent dans le même sens que l’action principale, c’est-à-dire amplifie la transformation, la rétroaction est dite positive
B. **Rétroaction négative** : si les nouvelles données produisent un résultat dans le sens contraire à celui fourni par les résultats précédents, la rétroaction est dite négative

Dans le chapitre suivant, nous allons exposer quelques méthodes d'apprentissage, notamment celles destinées aux agents réactifs et homogènes.
En premier lieu, nous allons nous intéresser à une technique d'apprentissage par l'exemple, à savoir les réseaux de neurones [37]. Comme son nom l'indique, cette technique est directement inspirée de l'étude du cerveau. Le principe général consiste à montrer une série d'exemples, chaque exemple étant associé à un stimulus d'entrée. Après cette phase d'apprentissage, si le réseau est stimulé avec une configuration d'entrées, alors il saura automatiquement de quel exemple il s'agit. Les réseaux de neurones trouvent de nombreuses applications dans de nombreux domaines, notamment en classification de données ou en reconnaissance vocale.

Ensuite, nous présenterons d'autres techniques d'auto-apprentissage. Ces méthodes, basées sur les processus markoviens, permettent notamment de construire un modèle d'apprentissage. Une fois ce modèle construit, la technique consiste à déterminer quelles sont les meilleures actions à réaliser. Lorsque ces actions permettent au système d'obtenir une récompense importante, leurs liens d'activation sont renforcés, on parle alors d'apprentissage par renforcement.

Et enfin, nous présenterons les algorithmes évolutionnistes. Inspirés de l'évolution darwinienne, le principe général consiste à attribuer à chaque individu une séquence génétique correspondant à un comportement. Régulièrement, les individus sont rassemblés pour pouvoir effectuer des croisements et des mutations sur leurs chaînes chromosomiques. L'apprentissage est donc découpé en générations successives et les performances de l'ensemble de la population sont ainsi accrues au fil de l'évolution. Bien sûr, toutes ces techniques d'apprentissage nécessitent un critère à optimiser, comment estimer la performance d'un individu au fil de l'évolution ? Ou encore comment récompenser un agent dans le cadre de l'auto-apprentissage ?

Chapitre II

TECHNIQUES D'APPRENTISSAGE POUR AGENTS REACTIFS ET HOMOGENES

2. Techniques d'apprentissage pour agents réactifs et homogènes

2.1 Introduction

Le principal avantage des systèmes Multi-agents réactifs est la simplicité des agents, ce qui rend leur conception plus aisée, mais le problème majeur reste la mise au point des différents paramètres des comportements réactifs pour produire le comportement global voulu.

La considération principale dans la conception des ces systèmes et le choix entre l'approche centralisée et l'approche décentralisée. La première approche nécessite la présence d'un agent de contrôle central qui programme à tout moment les tâches à effectuer par les autres agents. Initialement cette approche est facile à implémenter, mais avec l'augmentation de nombre d'agents le contrôle du système devient très difficile et très compliqué.

Dans l'approche décentralisée, chaque agent se comporte selon ses propres règles, il est autonome, ces règles lui permettent de communiquer avec les autres agents.

Si les agents se comportent selon les mêmes règles, la population est dite homogène sinon hétérogène.

Le choix entre l'approche centralisée et l'approche décentralisée dépend du travail à réaliser : pour les problèmes qui nécessitent un nombre limité et réduit d'agents le contrôle centralisé est la meilleure solution, par contre pour les travaux destinés à l'études des systèmes Multi-agents et les phénomènes d'auto organisation et d'adaptation le contrôle décentralisé est le mieux adapté.

On va s'intéresser aux systèmes multi-agents et les techniques d'apprentissage : au lieu de programmer un agent (robot) pour réaliser une tâche, nous allons le laisser apprendre seul sa propre stratégie. Les techniques de programmation des robots les plus répandues sont basées sur le tout programmé. C'est-à-dire que l'ensemble des situations envisageables a été programmé, soit de manière discrète, soit de manière continue, soit de manière mixte. La réaction d'un système préprogrammé face à une situation inconnue est incertaine. Il a donc fallu trouver des solutions alternatives à l'approche mathématique. C'est pour cette raison que nous allons s'intéresser aux techniques d'auto apprentissage.

L'introduction de ces techniques d'apprentissage est motivée par :

- La complexité des tâches à réaliser : interaction avec des environnements réels, complexes ou incertains ce qui rend difficile voir impossible la modélisation de ces environnement.
- L'autonomie et l'adaptabilité de ces systèmes

Les deux principaux freins dans leur réalisation sont :

- Les données mémorisées sont trop importantes : malgré l'accroissement des mémoires informatiques, certains problèmes ne sont pas solvables, faute de place mémoire.
- Le temps d'apprentissage dépend largement de la complexité du système.

Ces deux limites nous montre qu'il est préférable que l'apprentissage soit progressif : c'est-à-dire décomposer une tâche complexe en plusieurs taches de plus bas niveaux s'exécutant en parallèle.
Dans ce contexte nous allons détailler plus loin deux techniques d'apprentissage :

1 – apprentissage par renforcement (Travaux de Olivier Buffet [10])

2 – apprentissage par optimisation (Travaux de T.D Barfoot, G.M.T D'Eleuterio [11])

Nous précisons que les techniques d'apprentissage sont classées selon différents critères :

1 - Le domaine d'apprentissage

2 - Le type d'interaction avec l'environnement

3 - Les connaissances à priori disponibles

4 - La représentation des connaissances

2.2 Les réseaux neuronaux

2.2.1 Introduction et historique

En 1943, deux bio-physiciens de l'université de Chicago, Mac Cullogh et Pitts proposent le premier modèle de neurone biologique [18]. Ce neurone formel, aussi appelé neurone à seuil, est inspiré des récentes découvertes en biologie. Ce sont des neurones logiques (0 ou 1).

En 1949, le psychologue Donald Hebb introduit le terme connexionnisme pour parler de modèles massivement parallèles et connectés [19]. Il propose de nombreuses règles de mise à jour des poids dont la célèbre " règle de Hebb ".

En 1958, le psychologue Frank Rosenblatt, combinant les idées de ses prédécesseurs, propose le premier perceptron [20]. Ce réseau, capable d'apprendre à différencier des formes simples et à calculer certaines fonctions logiques, est inspiré du système visuel.

Au début des années 60, les travaux de Rosenblatt suscitent un vif enthousiasme dans le milieu scientifique. Mais en 1969, deux scientifiques américains de renom, Minsky et Papert, publient un livre [21] qui démontre les limites du perceptron proposé par Rosenblatt. En particulier, son incapacité à résoudre les problèmes non linéairement séparables, dont la fonction logique XOR est un célèbre exemple.

Les travaux ralentissent considérablement jusqu'aux années 80. En 1982, Hopfield démontre l'intérêt des réseaux entièrement connectés [23]. Parallèlement, Werbos conçoit un mécanisme d'apprentissage pour les réseaux multicouches de type perceptron : la rétro propagation (Back-Propagation). Cet algorithme, qui permet de propager l'erreur vers les couches cachées sera popularisé en 1986 dans un livre " Parallel Distributed Processing " par Rumelhart et al [22].
Depuis ces travaux, les applications des réseaux de neurones n'ont cessé de croître.

2.2.2 Notation

Le modèle général du neurone artificiel, qui est l'élément de base de beaucoup de réseaux, est composé des éléments suivants :

- une ou plusieurs entrées pondérées,
- un sommateur,
- une fonction de transfert,
- une sortie.

La Figure [2.4] montre le schéma général du neurone artificiel

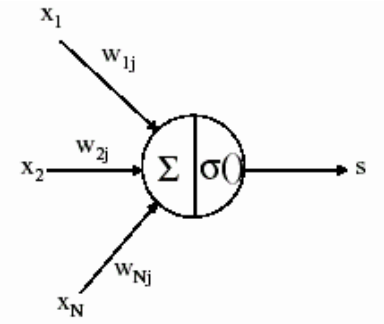


Figure 2.4 : ***Schéma général d'un neurone artificiel***

Avec :

- x_i le stimulus d'entrée,
- w_{ij} la valeur du poids synaptique reliant le stimulus i au neurone j,
- σ() la fonction de sortie du neurone,
- S la sortie du neurone.

La fonction d'un neurone artificiel est donnée par l'équation suivante :

$$S=\sigma\left(\sum_{j=1}^{n} x_j . w_{jt}\right) \quad \ldots\ldots \quad \textit{Equation 2.1}$$

Un réseau de neurones n'est finalement qu'une représentation conviviale de fonctions mathématiques. En effet, chaque réseau peut s'écrire sous la forme d'une équation. La fonction de transfert de base des réseaux est donnée par l'Equation 2.1. La fonction de sortie des neurones est principalement utilisée pour mettre en forme les signaux de sortie des neurones. Si par exemple le système est binaire, une fonction de type seuil pourra être utilisée (Figure 2.5). Si le système est borné alors la fonction de sortie peut aussi être bornée (Figure 2.6).

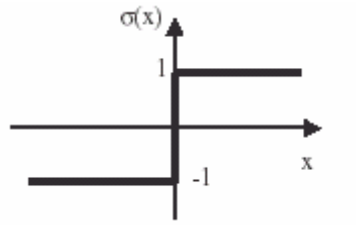


Figure 2.5 : ***Fonction de sortie de type seuil***

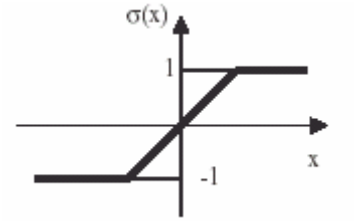


Figure 2.6 ***: Fonction de sortie bornée***

2.2.3 Conclusion

Cette méthode est supervisée, elle nécessite des exemples pour réaliser l'apprentissage, elle est donc difficilement applicable sinon impossible dans les cas d'auto-apprentissage et d'adaptation où aucun exemple de comportement n'est connu.

2.3 Apprentissage par renforcement (Reinforcement Learling)

2.3.1 Définition

Dans les systèmes multi-agents, l'apprentissage par renforcement a pour objectif de maximiser les performances du système en renforçant les meilleures actions. L'agent obtient des informations de l'état de l'environnement (perceptions) et agit sur l'environnement puis reçoit une estimation de sa performance : la récompense. Cette récompense peut être immédiate ou retardée. Le schéma général est représenté sur la figure suivante (Figure 2.7) :

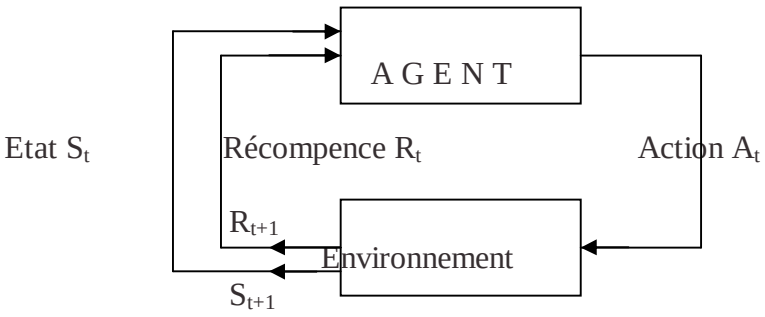


Figure 2.7 : ***Système d'apprentissage par renforcement***

L’agent connaît son état (S_t) dans l’environnement, il a la possibilité d’agir sur ce dernier (A_t) et il reçoit une récompense (R_t).

L’objectif de l’apprentissage par renforcement est d’associer à chaque état du système une action qui permet de maximiser la récompense, autrement dit chaque agent va apprendre localement son comportement de manière à optimiser une performance globale.

Les avantages de cette approche sont [38] :

- Un simple signal scalaire (la récompense) évaluant le comportement du système est suffisant pour apprendre, donc il n'est pas nécessaire de disposer d'un agent professeur connaissant à l'avance la solution du problème.
- Pour éviter une explosion combinatoire, chaque agent apprend son comportement réactif localement au lieu d'apprendre un comportement collectif du système.
- Les méthodes d'apprentissage par renforcement s'appuient sur le formalisme mathématique des processus décisionnels de Markov (PDM) qui précise les conditions de convergences des algorithmes.

Les difficultés rencontrées dans l'utilisation des méthodes d'apprentissage par renforcement décentralisé sont :

- l'agent n'a qu'une perception locale et partielle de l'environnement, il ne connaît pas l'état global du système dans lequel il se trouve
- une croissance exponentielle des calculs avec le nombre d'états et d'actions
- problème de distribution de la récompense : comment récompenser à leur juste valeur les actions effectuées.

D’où la proposition d'un apprentissage par renforcement décentralisé progressif qui, s'appuyant sur des techniques classiques d'apprentissage par renforcement, permet de doter les agents de comportements réactifs stochastiques. L'aspect progressif se fera selon deux axes, d'une part en confrontant les agents à des tâches de complexité croissante, d'autre part en augmentant progressivement le nombre d'agents.

2.3.2 Le Processus Décisionnel de Markov (PDM)

L'apprentissage par renforcement est basé sur l'apprentissage à temps discret des paramètres d'une chaîne de Markov. Les chaînes de Markov sont composées d'états et de transitions entre ces états.

Prenons par exemple le cas d'un agent qui peut se déplacer dans un environnement discrétisé (Figure 2.8). L'environnement est décomposé en 16 cases représentant chacune un état du

processus markovien. La position de départ est représentée par la case **S** et l'objectif est d'atteindre la case **E**. L'agent dispose de 4 actions possibles : se déplacer vers le nord, l'est, le sud ou l'ouest. Dans ce cas d'un système déterministe, lorsque l'agent se trouve dans un état donné S_n et qu'il réalise une action donnée A_t, l'état S_{n+1} suivant sera toujours le même. Dans le cas d'un système non déterministe, la même action réalisée depuis le même état S_n peut aboutir à des états S_{n+1} différents. Les transitions du processus markovien représentent la probabilité pour chaque état d'être atteint en fonction de l'action réalisée.

Le schéma général de l'apprentissage par renforcement consiste à associer à chaque action une fonction de cette probabilité et de la récompense associée à l'action [38]. La récompense moyenne espérée pour chaque action peut ainsi être déterminée. De cette façon, il ne reste plus qu'à exécuter l'action possédant la meilleure récompense espérée dans la phase d'application de la stratégie.

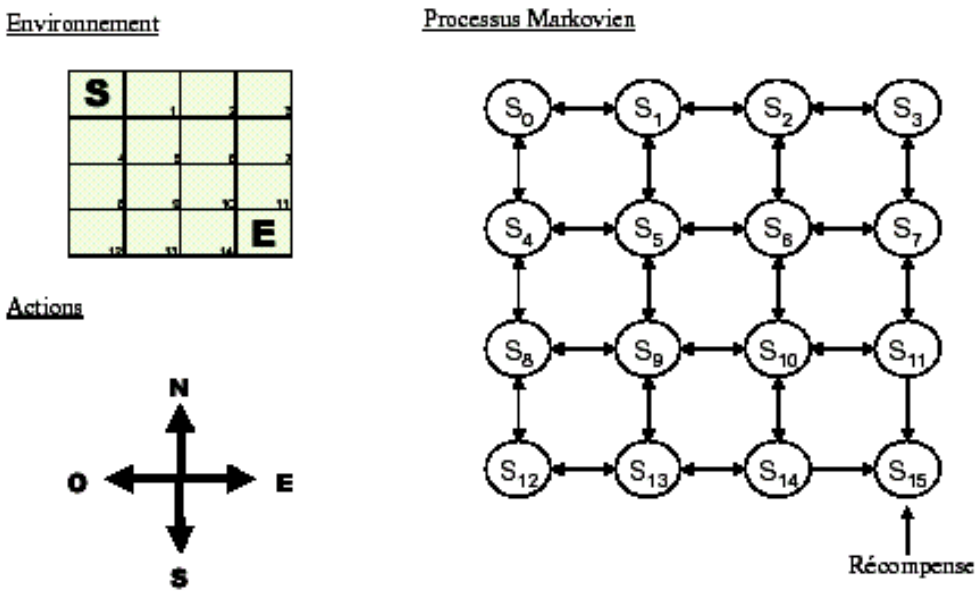


Figure 2.8 : ***Exemple de processus markovien***

La programmation dynamique s'applique pour les systèmes déterministes dont le modèle complet est connu, c'est-à-dire que tous les états et toutes les transitions sont connues. Mais en robotique, ces hypothèses déterministes sont rarement réunies. Il existe une autre méthode permettant de réaliser l'apprentissage avec une connaissance partielle du modèle : le Q-learning. Cet apprentissage permet également d'être appliqué aux systèmes non déterministes.

Nous commençons par détailler le cadre théorique idéal de l'apprentissage par renforcement, à savoir les processus décisionnels de Markov (PDM)

Un processus de décision markovien est défini par un tuple (S,A,T,R) dans lequel :

- S = {s} est un ensemble d'états,
- A = {a} est un ensemble d'actions,
- T : S x A x S **à** [0;1] est une fonction de transition exprimant la probabilité de passer entre les instants t et t+1 de l'état s à l'état s' sachant l'action choisie a comme suit : $T(s,a,s') = P(S_{t+1} = s' \setminus s_t = s, a_t = a)$.
  On écrit T(s,a,s') la probabilité de faire la transition de l'état s à l'état s' en utilisant l'action a.
- R : S x A x S **à** Π(R) une fonction indiquant la récompense reçue par le système après chaque transition d'état

T et R sont stationnaires : ne dépendent pas du temps

S et A sont deux ensembles finis.

Ce formalisme fournit les données d'apprentissage par renforcement, qui consiste à trouver un comportement qui est le plus performant possible. Ce comportement est décrit par une fonction π : S **à** Π(A) appelée politique π. Le problème à résoudre c'est la mesure de cette performance. La mesure de performance c'est l'espérance de gain si l'on suit la politique π à partir de l'état courant s, qu'on appelle l'utilité et qu'on note V.

$$V_p(s) = E\left(\sum_{t=0}^{\infty} g^t R_t\right) \quad où \quad g \in [0,1[$$

Typiquement, l'utilité peut être la somme des récompenses sur un horizon fini, ou la somme pondérée de cette récompense sur un horizon donné.

Cette formule calcule donc une espérance de gain en suivant une politique π et en notant R_t la récompense obtenue au pas du temps t (récompense pondérée par un coefficient g^t)

Tout PDM doit vérifier la propriété suivante : "la connaissance de l'état courant est suffisante pour savoir comment va évoluer le système, donc l'historique des états et actions passées étant la seule source d'autres informations". Cette propriété est traduite par la formule suivante :

$$\forall s' \in S \quad P(s_{t+1} = s' | s_t, a_t) = P(s_{t+1} = s' | s_t, a_t, s_{t-1}, a_{t-1}, ... s_, a_1, s_0, a_0)$$

2.3.3 Le Q-Learning : Proposé en 1989 par Watkins [8], le Q-learning est probablement la méthode d'apprentissage par renforcement la plus étudiée actuellement. Les seules hypothèses de départ sont les suivantes :

- le système est modélisable par une chaîne de Markov à états finis
- l'agent dispose d'un jeu d'actions discret.

Il n'est pas nécessaire de connaître les transitions entre les états, c'est le grand intérêt du Q-learning. L'apprentissage se fait par essai erreur. Contrairement à la programmation dynamique, l'objectif n'est pas d'estimer la fonction d'évaluation de chaque état, mais celle de chaque action, en fonction de l'état courant.

2.3.4 Apprentissage par renforcement et SMA

Les travaux de Boutilier [9] ont montré qu'un système multi-agent se modélise facilement sous la forme d'un PDM où l'état du système dépend de tous les agents et où une action est la composée des actions de tous les agents. Cette approche, est caractérisée par sa complexité exponentielle en nombre d'agents ce qui la rend irréaliste ainsi l'aspect décentralisé des SMA n'est pas pris en considération.

L'utilisation des méthodes classique d'apprentissage par renforcement pour résoudre un problème de manière décentralisée où chaque agent n'a qu'une vue partielle de l'état du système, pose des difficultés à savoir :

- La perte de stationnarité : du point de vue d'un agent, les autres agents sont des éléments imprévisibles de l'environnement, qui peuvent eux-mêmes apprendre. Donc la fonction de transition, du point de vue d'un agent, est non stationnaire : le résultat de son action dépend des autres agents, et eux même évoluent avec le temps.

  Supposons que chaque agent ait accès à l'état du système (observabilité totale) et que la loi de transition de l'environnement soit stationnaire. Un apprentissage centralisé correspondrait alors au cas d'un MDP classique. L'ensemble des agents pouvant être considéré comme une unique entité dont seuls les effecteurs sont distribués. Par contre, si chaque agent apprend son comportement de manière autonome, un agents quelconque verra tous ses congénères comme faisant partie de l'environnement. Or ces congénères sont aussi en phase d'apprentissage, donc en phase d'évolution de leur comportement, ce qui amène à une loi de transition non-sationnaire.

- Les perceptions partielles : un agent ne peut percevoir l'état global du système, ce qui est pourtant une hypothèse nécessaire pour assurer la convergence des algorithmes classiques vers un comportement optimal.

- La répartition de la récompense : deux approches sont proposées, soit tous les agents reçoivent la même récompense, autrement dit tous les agents ont accès à la même information sur la récompense commune, soit on associe à chaque agent une récompense à e qu'il perçoit de l'environnement.

2.3.5 Apprentissage progressif par renforcement (travaux d'olivier Buffet [10])

2.3.5.1 Problématique : étant donné un système multi-agent, composé d'agents qui apprennent individuellement leur propre comportement et ayant les caractéristiques principales suivantes :

- réactifs
- situés avec des perceptions partielles
- coopératifs : tous les agents partagent le même but et devront se coordonner pour y parvenir
- potentiellement hétérogènes : bien que disposant des mêmes capacités de perception, chaque agent apprend individuellement. Ainsi chaque agent peut acquérir un comportement différent des autres.
- La question posée est la suivante : "étant donnée une tâche globale, comment concevoir les comportements individuels des agents n'ayant qu'une vue locale et partielle de leur environnement ?"

2.3.5.2 Approche proposée

Vu l'absence des méthodes exactes permettant de trouver des comportement optimum pour les agents considérés, les méthodes approchées sont les plus conseillées. Cette approche est basée sur l'utilisation d'un algorithme d'apprentissage par renforcement permettant d'optimiser par une descente de gradient des politiques stochastiques.

Pour accélérer l'apprentissage et améliorer l'efficacité des comportements obtenus en évitant principalement que l'apprentissage ne s'arrête au premier optimum local rencontré, une méthodologie d'apprentissage progressif est proposée, en s'inspirant des travaux sur le "shaping" en psychologie.

L'idée principale de cette approche est de guider l'apprentissage en plaçant les agents dans des situations proches des sources (positives ou négatives) de renforcement, puis dans des

situations de plus en plus éloignées de ces sources de renforcement. Le fait de contraindre les agents à des situations simples permet de les faire coordonner plus rapidement. Petit à petit, en éloignant les agents de ces situations particulières, l'apprentissage devrait être favorisé grâce à ce que les agents ont déjà appris. L'apprentissage progressif se fait suivant les deux axes : une progression en complexité de la tâche à accomplir, puis une progression en nombre d'agents impliqués.

- **Progression en complexité** : On considère un nombre minimal d'agents mais suffisant à l'accomplissement de la tâche collective. Les agents sont placés d'abord dans des situations simples, puis dans des situations plus en plus complexes ce qui permet aux agents d'avoir plus de liberté de mouvement et d'explorer plus qu'avant leur environnement.
- **Progression en nombre** : L'apprentissage commence avec un faible nombre d'agents, chacun apprenant son propre comportement, ensuite d'autres agents sont ajoutés. Les comportements de tous les agents sont encore améliorés par apprentissage.

2.3.5.3 Validation expérimentale : Fusion de cubes

La tâche choisie met en œuvre des agents de deux types, jaunes ou bleus dans un monde pavé plan. Leur but est de pousser des cubes jaunes contre des cubes bleus. Quand deux agents coordonnent leurs mouvements pour atteindre ce but, les deux cubes disparaissent temporairement pour réapparaître de manière aléatoire ailleurs sur pavage, les agents responsables de cette fusion reçoivent une récompense positive. Les agents doivent provoquer de la fusion le plus souvent possible (Figure 2.9)..

Figure 2.9 : ***Un exemple de fusion de blocs***

Description des agents :

- **actions** : les agents disposent de quatre actions :
  - § déplacement d'une case vers le nord
  - § déplacement d'une case vers le sud

§ déplacement d'une case vers l'est

§ déplacement d'une case vers l'ouest

- **perceptions** : les perceptions de l'agent sont constituées des informations suivantes :

§ dir (aco) : la direction de l'agent de couleur opposée le plus proche parmi les quatre

§ directions (N-O-S-E)

§ pour le bloc jaune le plus proche :

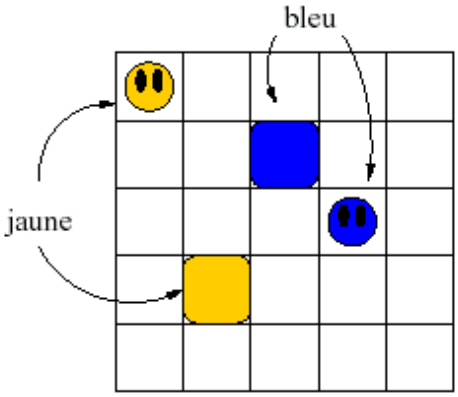
- dir (bj) : sa direction (N-NO-O-SO-S-SE-E-NE)

- proche(bj) : est-il sur les huit cases entourant l'agent (vrai / faux) ?

§ pour le bloc bleu le plus proche :

- dir (bb) : sa direction (N-NO-O-SO-S-SE-E-NE)

- proche(bj) : est-il sur les huit cases entourant l'agent (vrai / faux)?



agent	Dir(aco)	dir(bj)	dir(bb)	proche(bj)	proche(bb)
jaune	SE	S	E	non	Non
bleu	NO	O	NO	Non	Non

aco : agent de couleur opposé - bj : bloc jaune - bb : bloc bleu

Figure 2.10 ***: Exemple de perceptions*** *(deux agents dans un monde simple)*

En combinant ces perceptions, on arrive à un maximum de 1024 observations qui sera réduit à 256 en tenant compte des cas impossibles et des symétries qui loin du chiffre 15.249.024 représentant le nombre des états du problème centralisé et complètement observé dans un environnement de taille 8 x 8.

- récompense : chaque agent participant à la fusion de deux cubes reçoit une récompense positive (+5)

Voici un exemple de comportement des agents :

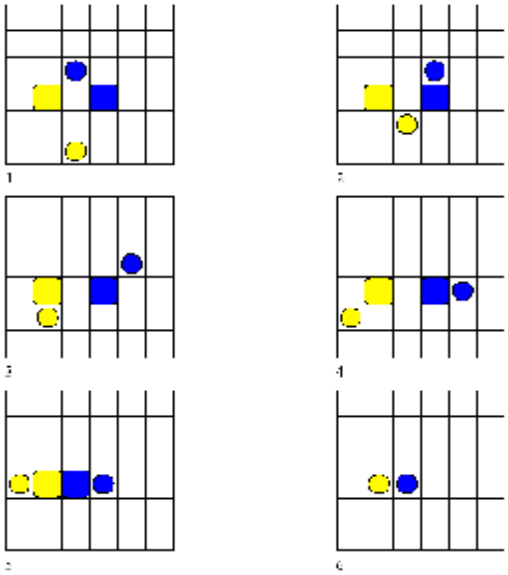


Figure 2.11 : ***Exemple de comportement d'agents***

Nous précisons que les agents apprennent à l'aide de l'algorithme de montée de gradient.

Progression en complexité :

L'apprentissage progressif en complexité est utilisé sur un dispositif réduit au minimum utile : un agent et un cube de chaque couleur. La table [2.1] montre une séquence d'expériences utilisées pour aider les agents dans leur entraînement

Situation initiale							
n (pas)	6	6	10	20	20	100	100
N (essais)	150	100	150	150	150	15	15

Tableau 2.1 : ***la séquence d'apprentissage utilisée pour l'apprentissage progressif***

Progression en nombre :

L'idée principale se fonde sur le fait qu'un comportement déjà appris dans une situation peut être un bon point de départ pour d'autres situations comparables.

Le principe est de réutiliser et adapter des agents dont le comportement a été appris dans une situation de deux agents et deux cubes (2a2c) dans des environnements plus encombrés, il peut être intéressant de s'en servir de base pour adapter les agents à cette nouvelle situation en prolongeant l'apprentissage, ce qui constitue cette apprentissage progressif en nombre.

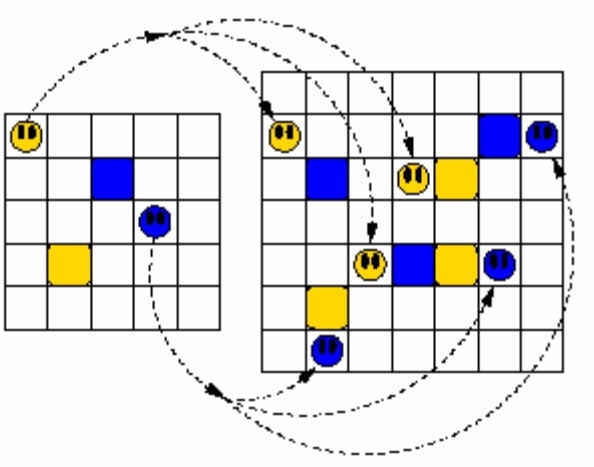


Figure 2.12 - ***Réplication des comportements de deux agents dans n agents***

2.3.5.4 Discussion :

Les méthodes classiques d'apprentissage par renforcement sont mal indiquées pour l'apprentissage de comportement réactif. Ces méthodes sont basées sur les processus Markoviens et demandent de discrétiser l'environnement en états, ainsi il est impossible de stocker une grande quantité d'information sur un système embarqué, ainsi que le temps d'apprentissage serait extrêmement long car l'agent doit visiter au moins une fois chaque combinaison état/action possible. L'idée est de proposer une nouvelle méthodologie d'apprentissage par renforcement dans un système multi-agents composé d'agents réactifs et coopérants. Chaque agent doit adapter individuellement son comportement local afin de réaliser une tache globale. Théoriquement ce problème d'apprentissage est sans solutions à cause de son aspect décentralisé et de la perception partielle du système par les agents apprenants. Cette nouvelle méthodologie d'apprentissage progressif, aussi bien au niveau de la complexité de la tâche que du nombre d'agents permet de pallier ces difficultés, Car on a montré qu'il est plus facile d'apprendre une tâche complexe de manière progressive que d'apprendre cette tâche directement. Mais le problème délicat de l'apprentissage progressif : comment l'agent peut-il utiliser les informations contenues dans la récompense qu'il reçoit pour décider de lui-même de la progression de son apprentissage ?

2.4 Apprentissage par optimisation

2.4.1 Introduction

Les chercheurs dans la robotique collective s'intéressent beaucoup à l'étude des sociétés d'insectes pour expliquer la motivation derrière leur travail. On a constaté que dans l'absence

d’un agent de contrôle central, les colonies de fourmis sont capables de travailler ensemble pour l’intérêt de la société entière chaque fourmis se comporte selon sa situation locale selon des règles de comportement très simples, ainsi un comportement global cohérant et intéressant en résulte. Il n’existe pas de fourmis particulières ou essentielles pour la dynamique totale du système.

La robotique collective a deux intérêts :

- aider à la conception des systèmes distribués
- avoir plus de connaissances sur la coordination au sein de collectivités biologiques

T.D. Barfoot et G.M.T d’Eleuterion [11] ont proposé une nouvelle approche d’apprentissage décentralisée pour un groupe de robots mobiles, homogènes et autonomes. Un système de contrôle identique est associé à chaque robot et constitué d’un automate cellulaire qui sert à arbitrer entre différents comportements de base.

L’introduction des algorithmes génétiques a permet de trouver l’automate cellulaire permettant de réaliser avec succès la tâche prédéfinie.

Avant de détailler cette approche, nous allons traiter quelques concepts de base nécessaire pour sa conception, a savoir : les algorithmes évolutionnaires et les automates cellulaires.

2.4.2 Les algorithmes évolutionnaires

Les algorithmes évolutionnaires sont des outils stochastiques de maximisation d'une fonction d'évaluation sur un espace de recherche, dont les éléments sont appelés individus. Un sous ensemble de l'espace de recherche, appelé population, évolue sur plusieurs générations. A chaque génération sont évalués par une fonction appelée fonction d'évaluation (fitness). La population est soumise à des opérateurs de sélection, de mutation et de croisement qui produisent une nouvelle population. Cette méthode est inspirée du principe d'évolution des espèces.

Ce principe est illustré par la figure [2.13] Marc Shoenauer [24]

Un algorithme évolutionnaire nécessite une représentation des individus qui sont les solutions du problème qu'on cherche à résoudre. La représentation est basée sur le codage binaire de l'information. La difficulté consiste à choisir le codage approprié.

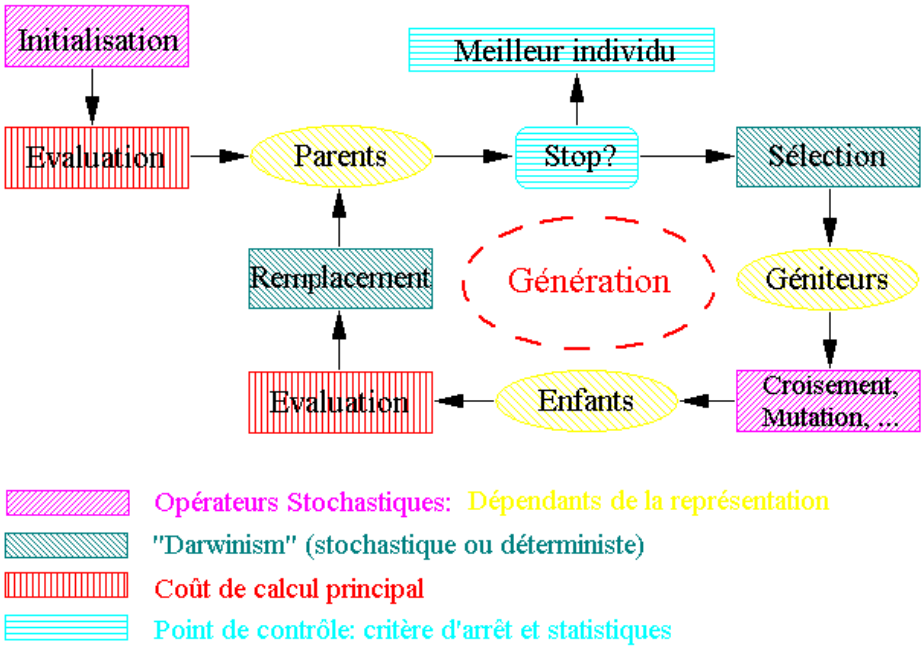


Figure 2.13 : ***Diagramme d'algorithme évolutionnaire***

Initialisation : l'initialisation sert à constituer la population initiale. C'est une étape très importante car si la population n'est par uniformément répartis au départ, l'évolution risque de se concentrer sur un optimum local.

Evaluation : Pendant les étapes d'évaluation, la fonction d'évaluation est calculée pour chaque individu, le but est d'évaluer chaque individu par rapport à un problème donné. Les individus ayant la valeur de la fitness la plus élevée seront appelés les meilleurs individus le choix d'une fonction d'évaluation appropriée au problème est critique.

Création de nouveaux individus enfants : Les nouveaux individus crées sont appelés les enfants, et les individus qui les ont engendrés sont appelés les parents. La création des enfants s'appuis sur les opérateurs de sélection, de croisement et de mutation

L'opérateur de sélection détermine des couples de parents parmi la population. A partir de ces couples, des enfants sont crées grâce à l'opérateur de croisement. Les enfants sont éventuellement mutés avec l'opérateur de mutation.

Le principe général de la méthode est sur l'algorithme suivant issu de [25]

1) Initialisation de la population P_i : remplir aléatoirement la chaîne chromosomique de chaque individu h_i.
2) Evaluer chaque individu de la population courante $F(h_i)$

3) Créer la génération suivante en sélectionnant statiquement les géniteurs. La probabilité qu'un individu soit utilisé pour générer la population suivante est donnée par :

$$P_r(h\ ) = \frac{F(h_i)}{\sum_{j=1}^{nbindividus} F(H_j)}$$

4) Réaliser les croisements sur les chaînes chromosomiques
5) Réaliser les mutations
6) Retourner en 2

Prenons pour exemple le problème du juste chiffre : nous disposons de quatre nombres et le but est de s'approcher le plus près possible d'un cinquième nombre en réalisant des opérations arithmétiques avec les 4 premiers. Nous supposerons que les opérateurs sont : l'addition, la multiplication, la soustraction et la division. Nous émettrons également comme hypothèse que l'opération arithmétique doit être constituée de trois opérateurs sans priorité (il n'y a pas de parenthèse dans l'opération). Par exemple, si les chiffres sont : 24 25 30 et 7 et que le résultat doit être 142, une bonne solution sera : 24 * 25 = 600 / 30 = 20 * 7 = 140. Si un algorithme génétique est utilisé pour résoudre ce problème, la chaîne chromosomique pourra être constituée de 14 bits : 8 bits pour les 4 nombres (Tableau 2.2) et 6 bits pour les 3 opérateurs (Tableau 2.3). Un exemple de chaîne chromosomique est donné sur le Tableau (2.4).

Codage	Nombres
00	24
01	24
10	30
11	7

Tableau 2.2 : ***Codage des chiffres dans la chaîne chromosomique***

Codage	Opérateurs
00	+
01	-
10	*
11	/

Tableau 2.3 : ***Codage des opérateurs dans la chaîne chromosomique***

0	1	0	1	0	0	0	0	1	0	1	0	1	1
25		-		24		+		30		*		7	

Tableau 2.4 : ***Exemple de chaîne chromosomique pour le problème du juste chiffre***

2.4.3 Les opérateurs génétiques

Les opérateurs génétiques sont utilisés pour générer la population suivante. On distingue deux types d'opérateurs : les opérateurs de croisement, qui réalisent une opération sur deux individus et les opérateurs de mutation, qui n'opèrent que sur un seul individu.

Les croisements : L'opérateur de croisement produit deux nouvelles chaînes à partir de deux chaînes initiales. Le masque de croisement est un mot binaire de même longueur que les chaînes génétiques. Ce masque contribue à déterminer lequel des deux parents sera géniteur de chaque bit de la génération suivante. Trois grandes familles de croisements sont distingués (figure 2.14)

Les croisements simple point. Les deux chaînes initiales vont être divisées en deux. La première partie de la première chaîne sera associée à la seconde partie de la seconde chaîne et inversement, deux nouveaux individus sont ainsi obtenus résultant d'un croisement entre les deux chaînes initiales.

Les croisements double points. Le principe est assez proche des croisements simple point, à cette différence qu'il y a deux points de séparation des chaînes, la chaîne initiale est divisée en 3 parties et la combinaison de ces 3 parties permet d'obtenir deux nouvelles chaînes.

Les croisements uniformes. Le masque de croisement est choisi aléatoirement, chaque bit peut être choisi indépendamment du reste de la chaîne.

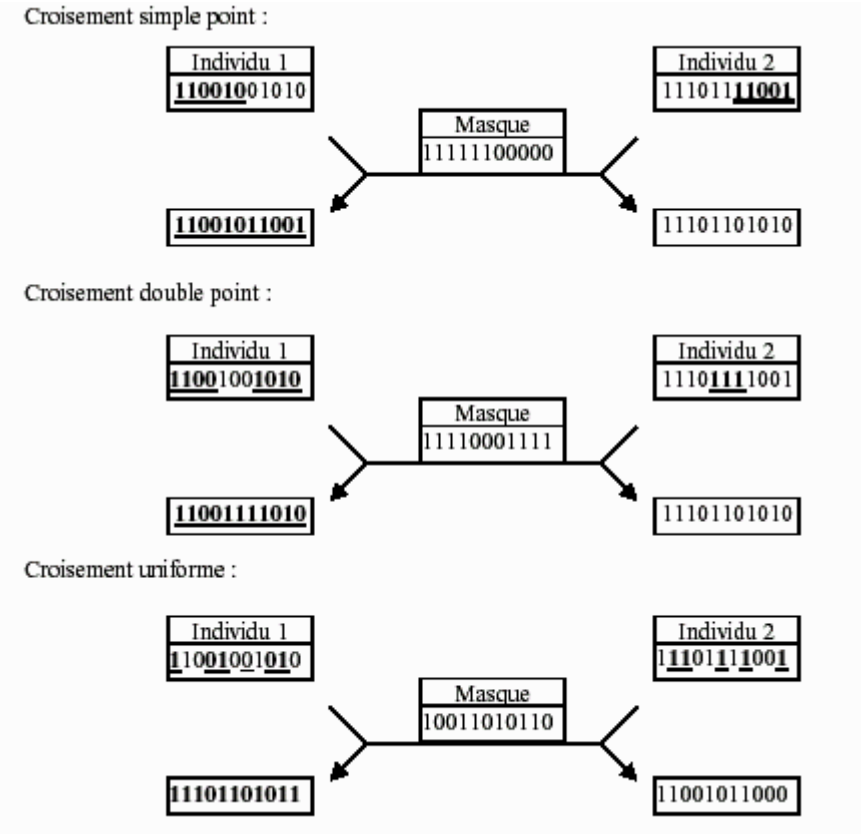


Figure 2.14 : ***Opérateurs de croisement***

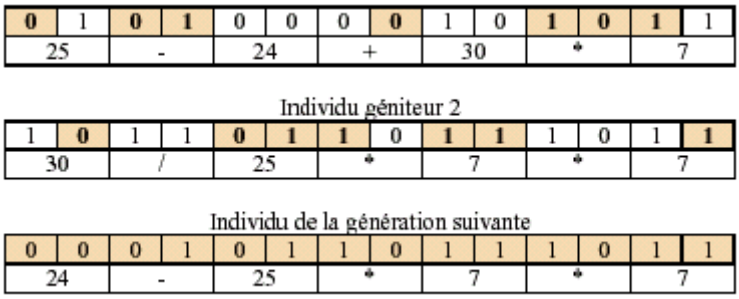


Figure 2.15 ***: Exemple de croisement pour le problème de juste chiffre***

Les mutations : Les mutations effectuent une modification à partir d'une seule chaîne initiale. Les mutations consistent à complémenter l'un des bits de la chaîne initiale. Les mutations permettent notamment d'obtenir des individus avec de nouvelles propriétés qui n'auraient pas été accessibles en ne réalisant que des croisements à partir de la population initiale (Figure 2.16)

Figure 2.16 : ***Exemple de mutation***

Dans l'exemple du juste chiffre présenté précédemment, un exemple de croisement est présenté sur la Figure 2.15. L'apprentissage a été réalisé sur une population de 20 individus. L'évolution du meilleur individu est montrée sur la Figure 2.17. On distingue que la solution optimale et la stabilité du système est atteinte après une quarantaine de générations. En réalisant l'apprentissage plusieurs fois, on trouve 4 résultats équivalents (L'objectif était d'atteindre le chiffre 142) :

((7 * 25) – 7) – 25 = 143
((25 * 7) – 7) – 25 = 143
((7 * 25) – 25) – 7 = 143
((25 * 7) – 25) – 7 = 143

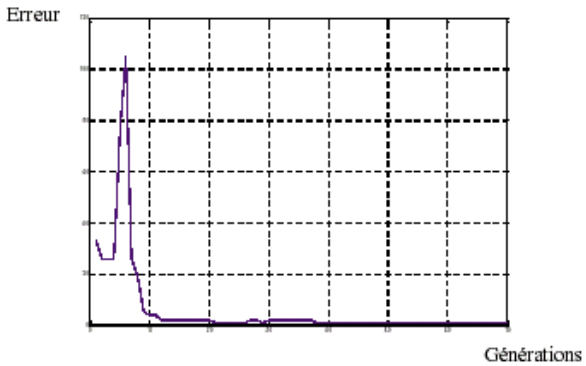


Figure 2.17 : ***Evolution du meilleur individu au fil des génération***

2.4.4 Domaines d'utilisation des algorithmes évolutionnaires

Les algorithmes évolutionnaires ont été appliques à différents problèmes issus de la recherche opérationnelle : le problème de voyageur de commerce, l'emploi du tempe, etc.

On trouve aussi des applications en robotique pour la recherche de trajectoire optimal ou l'évitement d'obstacles, en vision pour la détection ou la reconnaissance d'images, en recalage d'images médicales, en reconnaissance des forme, mais aussi en mécanique pour l'optimisation de forme, en économie et gestion de production, gestion du trafic aérien, etc. Ainsi ils ont été utilisés dans le cadre d'une recherche d'automates cellulaires ayant certaines propriétés par Mitchell et Crutchfreld.

2.5 Estimation de la performance

L'estimation de la performance appelée différemment selon les méthodes : la récompense pour le Q-Learning, l'aptitude pour les algorithmes génétiques ou encore l'erreur pour les réseaux de neurones artificiels. L'estimation de la performance est un paramètre essentiel de l'apprentissage. Le principal intérêt de l'apprentissage est de laisser l'agent trouver par lui-même la meilleure stratégie pour maximiser la récompense. Si la récompense est trop contrainte, l'apprentissage ne présente aucun avantage par rapport aux méthodes traditionnelles.

Les différents types d'estimations de la performance sont classés en 4 catégories :

1) *L'erreur* : utilisée dans les réseaux de neurones, l'erreur donne une information proportionnelle à la différence entre la sortie désirée et la sortie obtenue. Cette estimation de la performance est la plus contraignante car elle exige un modèle de comportement connu et fait tendre l'apprentissage à copier ce modèle.
2) *Les récompenses instantanées* : Les récompenses instantanées donnent une estimation de la performance correspondant à la stratégie courante. Contrairement à l'erreur, même si la sortie du système est un vecteur, la récompense instantanée est un paramètre unidimensionnel.
3) *Le coût* : La plupart des méthodes d'apprentissage sont présentées comme cherchant à maximiser la récompense. L'objectif de la tâche peut aussi être de minimiser un critère, par exemple une consommation d'énergie ou un temps de déplacement. Dans ces cas, on parle de coût et non de récompense. Mathématiquement, minimiser une fonction de coût peut toujours se ramener à maximiser une récompense.

4) *Les récompenses retardées* : Une récompense retardée est attribuée à la fin d'une séquence d'actions. Elle estime la performance de l'ensemble de ces actions. Ce type de récompense est parmi les moins contraignants.

2.6 Les automates cellulaires

Les automates cellulaires sont des systèmes complexes utilisés dans plusieurs simulations notamment en chimie, biologie, économie, etc.… On distingue les automates cellulaires selon leur espace d'évolution.

2.6.1 Automates cellulaires à une dimension

2.6.1.1 Définition

Un automate à une dimension 1D est un système évoluant dans tableau à une dimension appelé univers qui peut être infini ou borné. Chaque élément est appelé cellule peut prendre n état ou n est un entier positif supérieur on égal à 2. Les cellules ont un état qui peut changer au cours du temps. Le temps est discrétisé, l'unité de temps s'appelle génération.

A l'image d'organismes vivants, les cellules évoluent génération après génération. Cette évolution est déterminée par une règle transition locale. Cette règle de transition prend en compte le voisinage d'une cellule pour déterminer son état à la génération suivante. Autrement dit la valeur d'un cellule à la génération i+1 est déterminée à partir de sa valeur et des valeurs de ses 2 voisines à la génération i (Figure 2.18)

111	**110**	**101**	**100**	**011**	**010**	**001**	**000**
0	**1**	**0**	**1**	**1**	**0**	**1**	**0**

Figure 2.18 ***Représentation d'une règle d'un automate à une dimension***

2.6.1.2 Formalisation

Wolfram [26] introduit pour les automates de dimension 1 la notation suivant :

$$a_i^{(t)} = \mathrm{F}\left[a_{i-r}^{(t-1)}, a_{i-r+1}^{(t-1)}, ..., a_i^{(t-1)}, ..., a_{i+r}^{(t-1)}\right]$$

avec : $a_i^{(t)}$: valeur de la cellule i à la génération t.

F : règle de transition de l'automate.

r : rayon de la règle.

Afin de pouvoir identifier une règle, Wolfram propose de coder chaque règle par un nombre appelé "rule number".

2.6.1.3 Classes d'automates

Wolfram [27] a montré que les automates cellulaires à une dimension peuvent présenter un large spectre de comportements dynamiques, il introduit une classification en comparant leur comportement à celui de certains systèmes dynamiques continus. Il spécifie 4 classes d'automates cellulaires :

Classe 1 : Pour toutes les configurations initiales, les automates de cette classe évoluent après un temps fini vers un état homogène où toutes les cellules ont la même valeur.

Classe 2 : Les automates engendrent des structures simples dans le quelles quelques formes tables ou périodique survient.

Classe 3 : Les automates évoluent vers des comportements chaotiques.

Classe 4 : Les automates aboutissent à l'émergence de structures globales complexes.

2.6.2 Automate cellulaire à deux dimensions

2.6.2.1 Définition

A l'instar des automates 1D, le voisinage d'une cellule est utilisé pour déterminer l'état de cette cellule à la génération suivante de l'automate. Dans un espace 2D on peut différencier plusieurs types de voisinage possibles : voisinage de Moore, de Von Neumann et le voisinage diagonal.(figure 2.19).

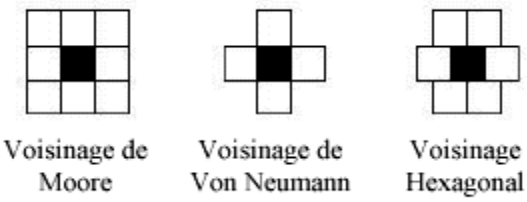


Figure 2.19 : ***Différents voisinages possibles dans un automate à 2 dimension*****s**

Wolfram a formalisé les règles de l'espace E des automates 2D à deux états en prenant en compte le voisinage de Moore [28] :

$$a_{(i,j)}^{(t)} = \mathrm{F}\left[a_{i-1,\,j-1}^{(t-1)}, a_{i-1,\,j}^{(t-1)}, a_{i+1,\,j}^{(t-1)}, a_{i,\,j-1}^{(t-1)}, a_{i,\,j}^{(t-1)}, a_{i,\,j+1}^{(t-1)}, a_{i+1,\,j-1}^{(t-1)}, a_{i+1,\,j}^{(t-1)}, a_{i+1,\,j+1}^{(t-1)}\right]$$

avec : $a_{(i,j)}^{(t)}$: valeur de la cellule de coordonnées (i,j) à la génération t.

F : règle de transition de l'automate.

L'espace E, comporte 2^{512} automates.

2.6.2.2 Sous-espaces d'automates

On distingue plusieurs sous-espaces dans l'espace E.

Sous-espace invariant par rotation : c'est un sous espace d'automates dans lequel la règle de transition d'un automate suit la propriété suivante :

∀ a, b, c, d, e, f, g, h et i appartenant à {0, 1} :

$$F\begin{bmatrix} a, b, c, \\ d, e, f, \\ g, h, i \end{bmatrix} = F\begin{bmatrix} g, d, a, \\ h, e, b, \\ i, f, c \end{bmatrix} = F\begin{bmatrix} i, h, g, \\ f, e, d, \\ c, b, a \end{bmatrix} = F\begin{bmatrix} c, f, i, \\ b, e, h, \\ a, d, g \end{bmatrix}$$

Cela signifie que deux cellules ayant deux voisinages différents mais identique après une rotation de $\frac{p}{2}, p$ ou $\frac{3p}{2}$ doivent avoir la même valeur à la génération suivante de l'automate. Le sous-espace d'automates invariants par rotation comporte 2^{140} automates.

Sous espace invariant par symétrie :

Un sous espace invariant par symétrie suit la propriété suivante :

$$F\begin{bmatrix} a, b, c, \\ d, e, f, \\ g, h, i \end{bmatrix} = F\begin{bmatrix} c, b, a, \\ f, e, d, \\ i, h, g \end{bmatrix} = F\begin{bmatrix} g, h, i, \\ d, e, f, \\ a, b, c \end{bmatrix} = F\begin{bmatrix} a, d, g, \\ b, e, h, \\ c, f, i \end{bmatrix} = F\begin{bmatrix} i, t, c, \\ h, e, b, \\ g, d, a \end{bmatrix}$$

Cela signifie que deux cellules ayant deux voisinages différents mais identiques à une symétrie centrale ou axiale doivent avoir la même valeur à la génération suivante de l'automate.

Le sous espace invariant par symétrie comporte 2^{288} automates.

Sous espaces complètement invariant : Les règles de transition vérifie les propriétés décrites ci-dessus. Cet espace comporte 2^{102} automates.

2.6.2.3 Automates isotropes

2.6.2.3.1 Notion de contexte

Le contexte d'une cellule est une matrice carrée de dimension 3 dont les éléments, sont les états de cette cellule et de ses 8 voisines (Table 2.5).

A(-1,1)	A(0,1)	A(1,1)
A(-1,0)	A(0,0)	A(1,0)
A(-1,-1)	A(0,-1)	A(1,-1)

Table 2.5 ***: Représentation du contexte d'une cellule****.*

Une cellule peut donc avoir 512 contextes différents.

2.6.2.3.2 Contextes symétriquement équivalents

Définition : Deux contextes A et B sont symétriquement équivalents si il existe n tel que quelque soit i, et j, A(i,j) est égal au $n^{ème}$ élément de la suite (B(i,j), B(-i,j), B(i,-j), B(-i, -j), B(j, -i), B(-j, i), (B-j,i), B(-j, -i).

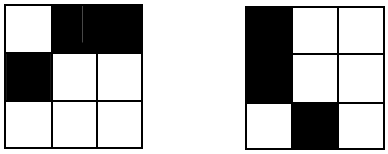


Figure 2.20 : ***Deux contextes symétriquement équivalents***

Dans cet exemple, les seuls éléments égaux à 1 sont A(1, 1), A(0,1),A(-1,0), B(-1,1), B(-1,0) et B(0,-1).

On a donc :

A(1,1) = B(-1,1).

A(0,1) = B(-1,0).

A(-1,0) = B(0,-1).

On en déduit que quelque soit i et j : A(i,j) = B(-j,i).

Ces deux contextes vérifiant donc la définition de contextes symétriquement équivalents avec une valeur n égal à 6. 6 étant le rang de B(-j,i) dans la liste (B(i,j), B(-i,j), B(i,j), B(-i,j), B(j,-i), B(j,i), B(-j,-i)).

Classe de contextes symétriquement équivalents : est un ensemble de contexte comportant tous les contextes symétriquement équivalent à l'un des contextes de l'ensemble.

Propriété : Tous les éléments d'une classe de contexte symétriquement équivalents sont symétriquement équivalents deux à deux.

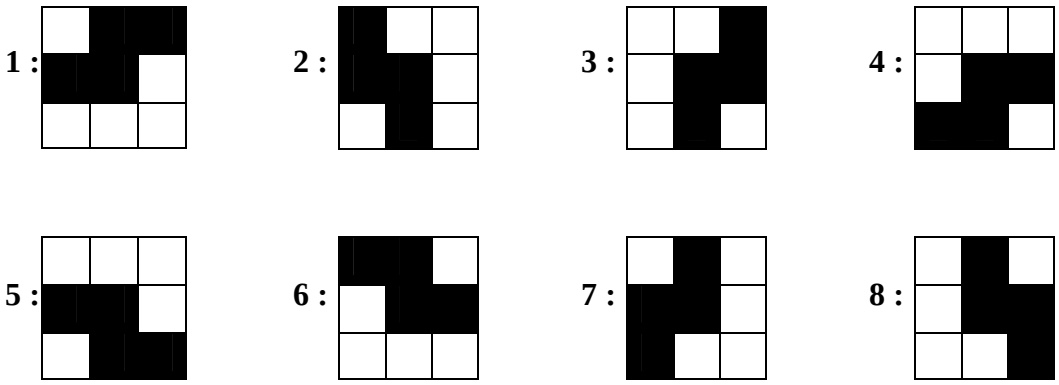


Figure 2.21 : ***Une classe de contexte S symétriquement équivalents***

Il y a 36 classes symétriquement équivalentes comportant 8 éléments (Figure 2.22).

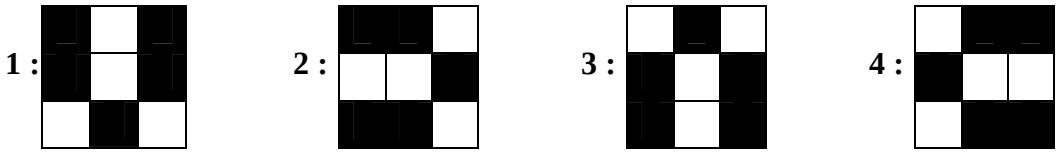
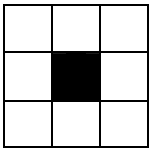


Figure 2.22 : ***Une classe de contexte S symétriquement équivalents***

Il y a 50 classes symétriquement équivalentes comportant 4 éléments (Figure 2.23).

Figure 2.23 : ***Une classe de contexte S symétriquement équivalents***

Il y a 8 classes symétriquement équivalentes Comportant deux éléments (Figure 2.24).



Classe de taille 1

Figure 2.24 : ***Une classe de contexte S symétriquement équivalents***

Il y a 8 classes de contextes symétriquement équivalents Comportant un seul élément.

Donc sur les 512 contextes possibles, 8 appartiennent à une classe d'un élément, 16 appartiennent à une classe de deux éléments, 200 appartiennent à une classe de 4 éléments et 288 appartiennent à une classe de 8 éléments.

Le nombre total des classes est 102.

L'introduction de cette notion d'équivalence permet de simplifier le nombre d'état d'un automate cellulaire à deux dimensions.

2.7 Formation de tas par apprentissage par optimisation (Barfoot & d'Eleuterion [11])

2.7.1 Introduction

L'approche présentée est une approche décentralisée, les agents sont homogènes : tous les agents sont identiques et disposent des mêmes capacités de perception et d'action, ainsi la communication est implicite (l'agent agit seulement aux contraints physique).

Nous allons exposer l'architecture de base d'un agent ce qui représente la partie du système qui n'évolue pas. Ainsi que l'autre partie du système qui évolue avec le temps, sous forme d'automate cellulaire avec la description des algorithmes génétiques utilisés.

En deuxième lieu, nous allons présenter la tâche à réaliser qui nécessite une coordination globale (Formation de tas) (Clustering) [39], et les résultats des simulations réalisées.

On dispose donc d'un ensemble de robots (agents) et objets répartis sur un monde pavé plan, les agents ont une tâche commune, qui consiste à rassembler les objets sous forme d'un tas (cluster). Les agents n'ont pas information sur la façon de constituer ce tas, mais ils doivent coordonner leurs choix pour produire un comportement global cohérent

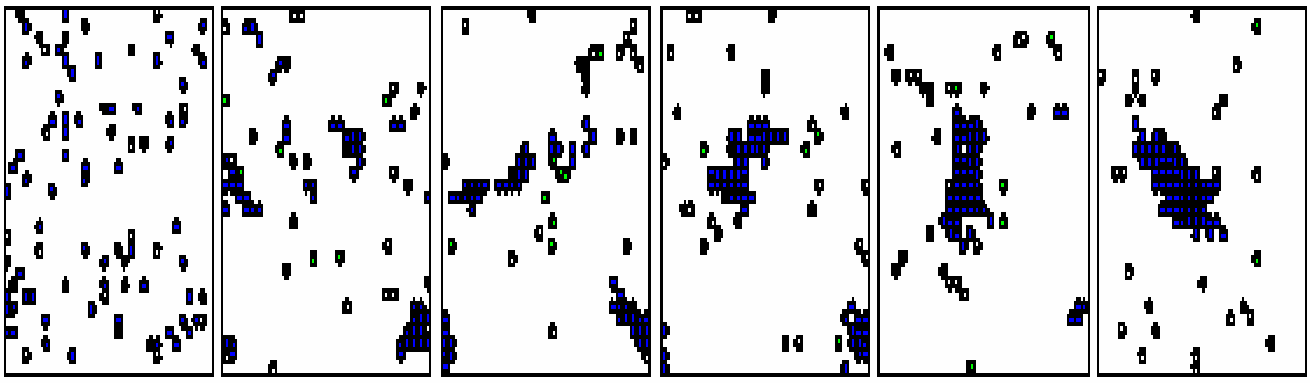


*Figure 2.25 : **Tache de "Clustering"** : les agents (cercle avec trait) doivent rassembler les objets (cercles noirs) à partir d'une distribution initiale aléatoire*

2.7.2 architecture d'un agent

Chaque robot (agent) possède un nombre de capteurs (Sensors) et d'effecteurs (Actuators). Une phase de prétraitement (preprocessing) consiste à transformer les informations des capteurs aux données discrètes d'entrée (Figure 2.26). Les sorties du système de contrôle d'un agent doivent être discrètes sous forme d'un ensemble de comportement de base (Mataric

1997). Par conséquent, une autre phase de post-traitement (Postprocessing) consiste à transformer les valeurs discrètes de sortie au système de control des effecteurs. Le défi est de trouver un schéma d'arbitration qui à partir d'une séquence discrète d'entrée (taille N), trouver la séquence discrète de sortie appropriée (un comportement parmi les M comportements de base). Ce schéma est une table de recherche (lookup table) similaire à un automate cellulaire (fig 2.27).

Pour chaque séquence d'entrée correspond une valeur discrète de sortie représentant un comportement de base.

Le robot cherche à tout le temps l'action correspondante à sa perception actuelle et l'exécute. La taille de la table est 2^N, le nombre de comportement de base M n'influe pas sur cette taille mais sur le nombre de toutes les tables possibles qui est $M^{(2n)}$.

L'étape cruciale dans toute cette approche est de trouver les tables de recherche qui permettent à un ensemble d'agents de réaliser avec succès leur tâche.

La méthode proposée est l'optimisation évolutionnaire (Evolutionary optimization).

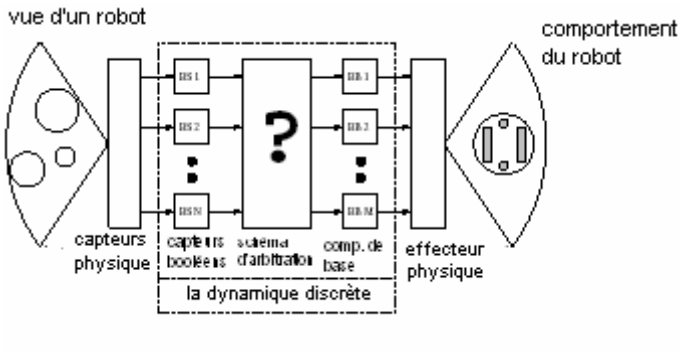


Figure 2.26 – ***Schéma de contrôle pour un robot***

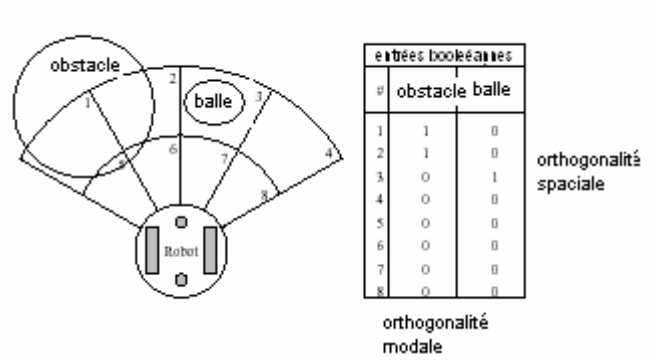


Figure 2.27 – ***Les perceptions d'un robot***

2.7.3 Principe de l'approche utilisée

Les algorithmes génétiques représentent une technique d'optimisation globale basée sur l'évolution biologique.

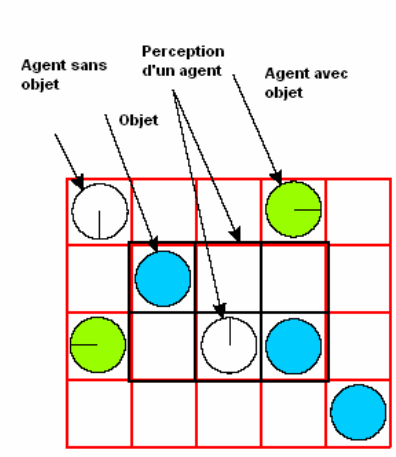
Une population initiale et aléatoire de P Table de recherche (Cellular automata lookup table) représentant chacune un automate cellulaire jouant le rôle d'arbitre dans un entraînement donné d'une collectivité de robots identiques (ont la même table de recherche). Ces P tables représentent génétiquement P chromosomes générés aléatoirement, constitués d'une séquence de toutes les valeurs discrètes de la table et qui seront évolués pendant G générations. Une fonction de performance est assignée à chaque table et qui sera calculée pour chaque génération. Cela permet de déterminer si elle peut faire partie de la prochaine génération Les opérateurs génétique de croisement et de mutation permettent l'introduction de nouveau tables de recherche dans la population, les meilleur tables de nombre k tel que $k \leq P$ sont gardées pour la génération suivante. Le reste (P-k) subi un croisement à partir d'une position aléatoire et avec une probabilité P_c, ainsi à des mutation avec une probabilité P_m. P_c et P_m sont choisie d'une distribution aléatoire uniforme.

2.7.4 Description de la tâche

On dispose d'un ensemble de robots répartis sur un espace bidimensionnel (figue 2.28), leur tâche à accomplir est de déplacer un ensemble d'objets distribués aléatoirement sur la même surface pour les rassembler sur un seul tas (cluster), et ceci dans l'absence d'un agent de contrôle central pour indiquer l'emplacement de ce « cluster ». Les robots doivent réaliser cette tache eux seuls et sans intervention externe d'où la complexité du système à réaliser.

Les robots sont situes dans espace physique divise en J cellules A_j comme illustré dans la figure (2.29). La fonction de performance (fitness) est définie par :

$$f_{total} = \frac{\sum_{i=1}^{I} f_i}{I}$$



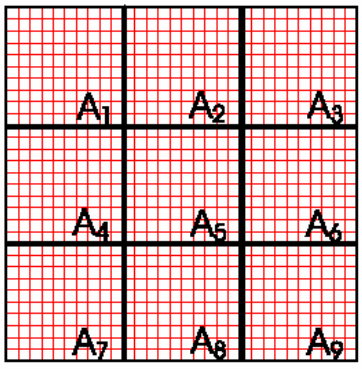


Fig 2.28 - ***Perceptions d'un agent*** *Fig 2.29 -* ***Espace physique divisé en cellules***

Avec I est le nombre de conditions Initiales aléatoire ; f_i est la fonction de performance de l'une des conditions initiales donnée par :

$$f_i = 1.0 + \frac{\sum_{j=1}^{J} p_j \ln p_j}{\ln p_j} \qquad \text{Où :} \qquad p_j = \frac{n(A_j)}{\sum_{j=1}^{J} n(A_j)}$$

Et $n(A_j)$ est le nombre d'objets dans la cellule A_j .

Cette fonction vaut 0 si tous les objets sont distribués d'une façon équitable sur touts les cellules et vaut 1 si tous les objets sont regroupés sur seule cellule.

La fonction fitness est assignée à chaque table de recherche (CA lookup table) à la fin d'une période de temps T, fi est calculée pour indiquer si les objets sont bien regroupés, ce processus est répété pendant I fois pour déterminer enfin f_{total} qui est la valeur de fitness de la table de recherche .

2.7.5 Simulation et résultat

Les agents sont situes dans un environnement sous forme d'un espace bidimensionnel divisé en cellules. A tout moment chaque agent perçoit 6 carrés. Il peut transporter un objet, ou informer les autres agents s'il est en possession ou non d'un objet (Figure 2.28)

Perception :

L'agent perçoit les cinq carrées qui l'entourent, on a trois possibilités :

- le carré contient un objet
- le carré contient un agent
- le carré est vide
- Pour le sixième carré occupé par l'agent lui-même on a deux possibilités : agent transporte un objet

  Agent ne transporte pas un objet

Donc on 3^5 x 2= 486 possibilités et donc 486 entrée dans la table de recherche.

Actions :

On distingue 2 comportements de base :

- déplacement en avant, gauche et à droite

- Manipulation d'objet : l'agent ramasse/ dépose un objet situé en avant (si possible) sinon ramasser/déposer l'objet à gauche si possible ou à droite sinon activer un comportement de déplacement

Avec ces 2 comportements, on s'est retrouvé avec une table de taille 486 et donc on a $2^{486} = 10^{146}$ tables de recherche possibles qui représente un nombre très grand, d'où la nécessité l'utilisation d'une approche évolutionnaire.

Dans cette expérience, on a utilisé un population de taille P=50, un nombre de génération G=150 une probabilité de croisement Pc= 0,6, une probabilité de mutation Pm=0,005 dans un espace de dimension 31x30 où résident 30 agents et 60 objets. Pendant un temps d'apprentissage T= 2000 pas de temps, nombre de surface nécessaires pour le calcul de la fitness J=9 et le nombre conditions initiales pour évaluer la fitness I= 30 (voir résultats obtenus dans figure 2.30)

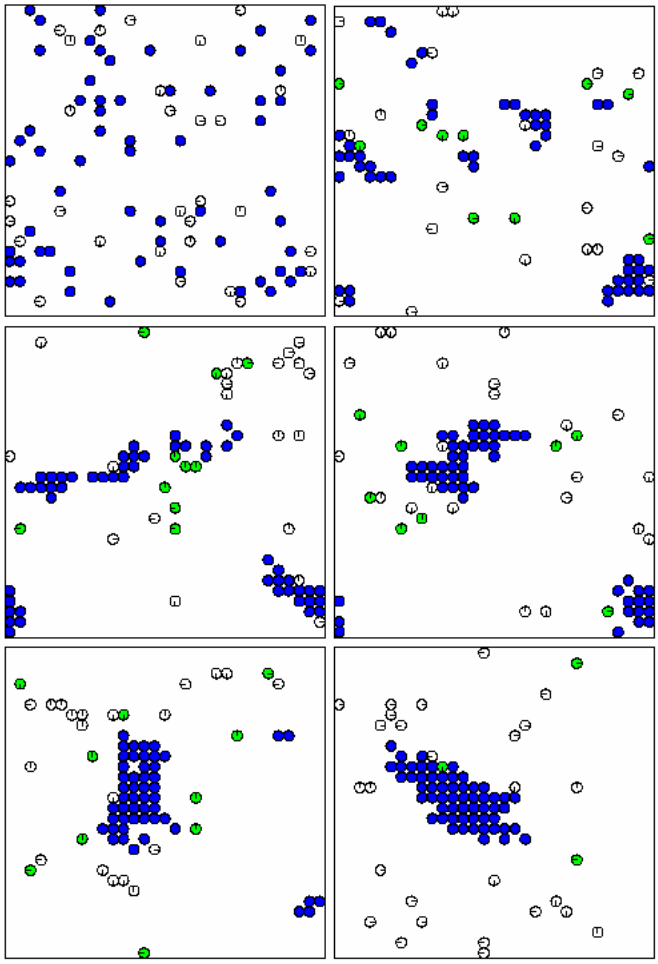


Figure : 2.30 Comportement des agents en fonction du temps d'apprentissage (0,197,570,1701,5039,9997) dans un monde de dimension (31x30)

Complexité en nombre (scaling the solution) : consiste à modifier la taille du problème en gardant la taille de la table de recherche inchangée, en faisant :

- Changer le nombre d'agents dans une simulation en gardant les autres paramètres inchangés.
- Changer le nombre d'objets dans une simulation en gardant les autres paramètres inchangés.
- Changer la taille du monde dans une simulation en gardant les autres paramètres inchangés.
- Changer le nombre d'agents, le nombre d'objets et la taille du monde dans une simulation en gardant les densités d'agents et d'objets inchangées.

Augmentation de nombre d'agents : La simulation est réalisée dans un monde de dimension (31x30) contenant 60 objets, la question posé : en faisant varier le nombre d'agents, quel est le nombre d'agent nécessaire pour réaliser la tâche voulue ? L'expérience a montré que la valeur optimale de fitness correspond à un nombre d'agent au voisinage de 30, et que l'utilisation d'un nombre élevé d'agents déconseillé ainsi qu'un nombre trop petit.

Augmentation de nombre d'objets : La simulation est réalisée dans un monde de dimension (31x30) et nombre fixe d'agents (30), la question posée est : quel est le nombre d'objets dont les agents sont capables à manipuler ? L'expérience a montré que moins d'objets engendrent plus de performance.

Augmentation de la taille du monde : pour un nombre fixe d'objets et d'agent, quel est l'impact de la variation de la taille du monde sur la réalisation de la tâche voulue.

Après ces différentes expériences, il s'est avéré que la solution idéale est de faire varier la taille du monde, le nombre d'agent et le nombre d'objet simultanément en gardant les densités des agent et des objets invariantes. Par exemple un agent par 30 carré et 1 objet par 15 carré.

2.7.6 Conclusion

Les travaux antérieurs ont montré la possibilité d'apprendre des comportements réactifs en utilisant les algorithmes génétiques, cette méthode est centralisée, elle nécessite de centraliser l'ensemble des chaînes chromosomiques afin de produire la génération suivante. Une panne du système central de contrôle engendre immédiatement une panne globale sur l'ensemble du système. L'application de cette méthode sur des systèmes réels a montré que le point le plus long

dans l'apprentissage ne sera pas la réalisation des croisements ou des mutations, mais l'estimation de la performance de la population.

L'idée proposée par T.D. Barfoot et G.M.T d'Eleuterion [11] est d'exploiter à fond le parallélisme des systèmes multi-agents afin de diminuer les durées d'apprentissage en distribuant l'élément le plus coûteux de l'apprentissage (en temps de calcul), c'est-à-dire l'estimation de la performance.

L'approche proposée a montré qu'une méthode d'optimisation évolutionnaire peut être utilisée pour découvrir les règles locales de comportement des agents qui vont produire le comportement global désiré. Donc au lieu de dessiner à la main les règles d'interaction et essayer de prédire les conséquences globales ou pis encore accepter toutes les conséquences, on a pré-spécifié le comportement global, en permettant aux agents d'évoluer leurs règles locales de comportement.

Le chapitre suivant est consacré à l'étude d'un nouveau paradigme permettant la conception et le développement des modèles de calcul autonome, et de résoudre les problèmes difficiles, cela en tenant en considération que l'autonomie est le noyau de modélisation des comportements dans n'importe quel système complexe et que chaque entité autonome interagit localement avec son voisinage.

Chapitre III

AUTONOMY ORIENTED COMPUTING A.O.C

3. Calcul orienté par l'autonomie
Autonomy Oriented Computing (A.O.C)

3.1 Introduction

Avec l'évolution des méthodes et outils de calcul, on s'intéresse actuellement à des problèmes de plus en plus complexes tels que les réseaux sans fil, la robotique collective, l'analyse du marché économique. Ces problèmes ont comme caractéristiques principales c'est que le calcul est distribué, et qu'il n'existe pas une machine unique dédiée pour accomplir cette tâche, ainsi le succès de ces applications est réalisé à travers le déploiement à grande échelle d'un grand nombre d'agents de calcul capables d'une manière autonome de prendre des décisions locales pour aboutir au but global désiré. D'où la nécessité d'introduire une méthodologie permettant de développer des modèles de calcul autonome, ainsi de résoudre les problèmes difficiles.

Deux Approches sont utilisées pour modéliser un système complexe [15]:

a – L'approche descendante (Top-Down) : on démarre à partir des caractéristiques du haut niveau du système complexe. Cette approche n'est pas toujours réalisable, car on ne peut pas toujours décrire le comportement émergent et dynamique d'un système complexe.

b – L'approche ascendante (Bottom-Up) : On démarre à partir de simples entités, en modélisant leurs comportements comme suit [14]:

- **Autonomie** : elles sont rationnelles et agissent d'une façon indépendante, ceci dans l'absence d'un contrôleur central ou d'un coordinateur.
- **Emergence** : elles exhibent un comportement complexe qui n'était pas présent ou prédéfini dans leurs comportements élémentaires et qui le résultat des interactions entre entités.
- **Adaptabilité** : elles changent leurs comportements en fonction du changement de l'environnement.
- **Auto organisation** : ces entités sont capable à s'auto organiser pour atteindre le comportement désiré.

Dans ce contexte nous allons introduire une approche ascendante appelée A.O.C Autonomy Oriented Computing comme nouveau paradigme permettant de modéliser les systèmes complexes et de résoudre les problèmes difficiles, ainsi pour caractériser les comportements des systèmes complexes.

3.2 Définition de A.O.C :

Le calcul orienté par l'autonomie (A.O.C) est une approche ascendante pour l'étude des systèmes complexes, introduit par jiming liu en 2001, l'objectif est de développer des algorithmes ou systèmes de calcul qui utilisent la notion d'autonomie comme noyau de modélisation de comportements dans n'importe quel système complexe, en s'inspirant des phénomène d'autonomie et d'auto-oganisation dans la nature [15]. Donc a deux objectifs à atteindre :

- Modélisation des systèmes complexes
- Résolution des problèmes de calcul très compliqués

L'introduction de cette approche est motivée par le besoin de résoudre des problèmes distribués à grande échelle ainsi la modélisation des systèmes complexes. D'où l'importance d'utilisation des algorithmes auto adaptatifs.

En d'autres termes, A.O.C permet de modéliser, expliquer et prédire les comportements dans un système complexe adaptatif.

A.O.C ayant rapport avec différents domaines de recherches tel que :

- la vie artificielle (Artificial life) : basé sur la simulation de la vie sur un ordinateur, par contre dans AOC on n'a pas besoin de reproduire d'une manière exacte un comportement naturel observé. Ce comportement sera soumis à un processus d'abstraction et de simplification.
- Simulation basée agent (Agent-based simulation ABS) : basé sur la reproduction de certains comportements complexes dans un contexte social et trouver éventuellement une explication du phénomène observé. A la différence de AOC, le comportement des agents n'a pas toujours de similaire dans la nature, donc plusieurs problèmes n'ont pas de solution avec ABS
- Systèmes multi-agents (Multiagent systems MAS) : dans ces systèmes la négociation et la coordination sont des concepts importants, d'où la nécessité d'intervention humaine.
- Systèmes SWARM : basé sur l'utilisation du métaphore d'une société d'insectes pour résoudre un problème.

3.3 Modélisation d'un processus A.O.C

Un système A.O.C est composé de (figure 3.31) :

- Un ensemble d'entités autonomes
- Un environnement

- Des interactions : soit entre entités et environnement, soit entre entités eux-mêmes
- Une fonction objectif du système.

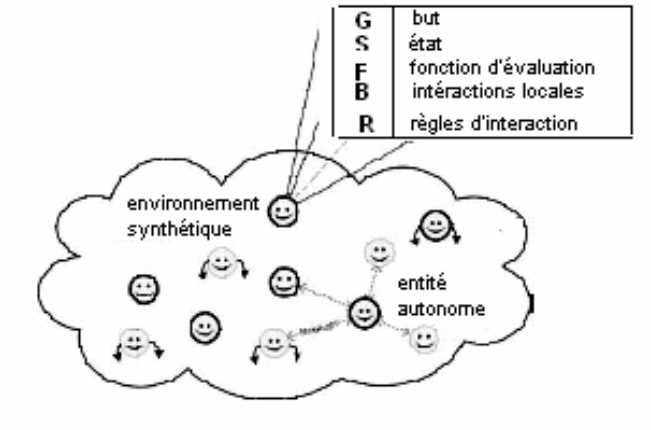


Figure : 3.31 ***La structure d'un système AOC***

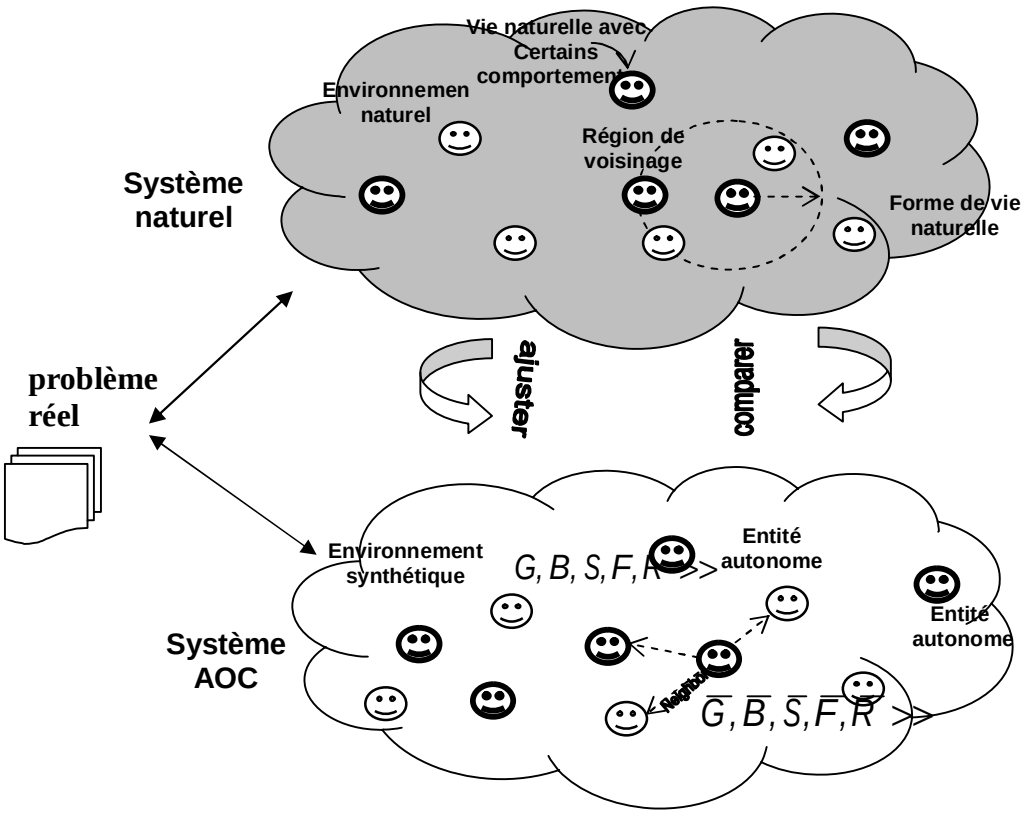
A.O.C a trois objectifs à atteindre [15] :

- Reproduire un comportement observé de la vie dans le calcul, et ceci après sa simplification
- Comprendre le mécanisme d'un système complexe réel en effectuant plusieurs expériences
- Emergence de la solution d'un problème dans l'absence d'une intervention humaine.

Pour construire un modèle basé sur A.O.C, on suit les étapes suivantes (figure 3.32)

1. Découvrir un phénomène naturel ayant relation avec le problème à résoudre ou similaire au système à réaliser manuellement.
2. Observer le comportement macroscopique du système naturel
3. Observer les comportements simples des entités élémentaires du système naturel en particuliers leurs interactions. (entre eux ou avec l'environnement)
4. Construire un plan illustrant les correspondances entre certains états du système naturel et la solution manuelle du problème
5. Concevoir des entités synthétiques équipées de comportements élémentaires par analogie avec le système naturel.
6. Construire le système synthétique par analogie
7. Exécuter le système synthétique

8. Observer le comportement macroscopique du système synthétique, de même que le comportement microscopique des entités synthétiques.
9. Valider les comportements macroscopiques et microscopiques du système synthétique avec ceux du système naturel homologue.
10. Modifier 5-6 en vue d'avoir 8
11. Répéter les étapes 5-10 jusqu'à satisfaction.
12. Appliquer le modèle final pour résoudre un problème complexe



G	but	**B**	Comportement local	**R**	Régles de comp.
S	état	**F**	Fonction d'évaluation		

Figure 3.32 : ***Conception d'un système AOC***

Donc la formulation d'un modèle A.O.C nécessite 3 phase :

Phase 1 : Identification du système naturel : c'est trouver une analogie avec le monde naturel. Cette phase nécessite l'identification du comportement du système désiré et ses paramètres (nombre d'entités autonomes et leur durée de vie). Le choix d'une analogie correcte est la clé de succès d'un système basé sur A.O.C

Phase 2 : Construction du système artificiel, cette phase est divisée en deux :

- Modélisation des entités : par identification de ces entités, en définissant leurs caractéristiques (états, buts, …) et leur voisinage qui nécessite le calcul d'une certaine mesure (distance) permettant de définir la zone d'interaction entre les entités autonomes. Ainsi la définition du comportement local et les règles de comportement
- Modélisation de l'environnement : par identification de ces caractéristiques et définition de sa représentation.

Phase 3 : Mesure de performance : comprend un critère d'évaluation pour comparer le système artificiel modélisé par A.O.C avec son homologue naturel, ce qui provoque éventuellement une modification sur les comportements individuels et les règles de comportement. La fin de cette phase déclenche un nouveau cycle.

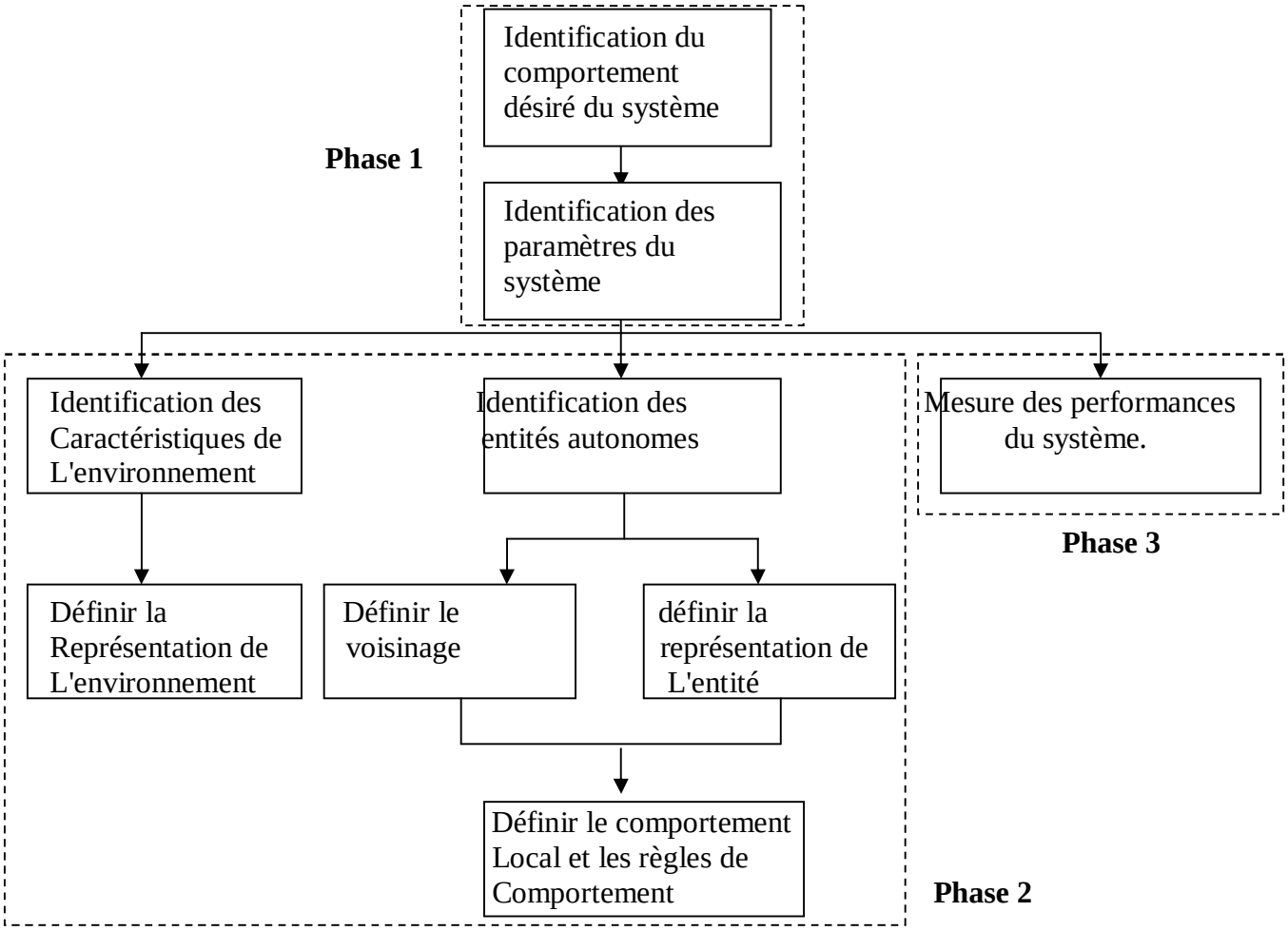


Figure 3.33 ***Modèle A.O.C***

3.4 Les approches de A.O.C

Le calcul orienté par l'autonomie A.O.C utilise la notion d'autonomie comme noyau de modélisation de comportements dans un système complexe, il permet de reconstruire, d'expliquer et de prédire le comportement de tel système. Les interactions entre les entités autonomes représentent la force motrice de A.O.C.

La formulation d'un système basé sur A.O.C nécessite une analogie avec un système naturel, cette analogie exige l'identification, l'abstraction et la reproduction de certains phénomènes naturels. Le processus d'abstraction peut nécessiter une simplification du système naturel homologue. Il existe 3 approches de A.O.C :

3.4.1 A.O.C par fabrication (AOC-by-fabrication) (figure 3.34) : est basé sur la duplication de certains comportements auto organisés observés dans la nature pour résoudre des problèmes complexes. La vie artificielle, les algorithmes génétiques et les colonies de fourmis sont des exemples typiques de cette approche. Dans les algorithmes génétiques (Genetic algorithm) le processus d'évolution est simplifié à la sélection, croisement et mutation sans tenir en considération la notion du sexe. La programmation évolutionnaire (Evolutionary programming) est basée sur le phénomène de reproduction et l'ajout d'une contrainte sur la mutation.
Les caractéristiques de cette approche sont :

a) On dispose d'une population d'individus (entités) caractérisés essentiellement par des ensembles de buts, d'états, de comportements et de règles de comportement, ces entités peuvent être homogènes ou hétérogènes.
b) La composition de la population peut être changée à chaque instant soit par augmentation ou par diminution.
c) Les interactions entre entités sont locales : pas d'informations globales ni de contrôleur central de comportements est nécessaire.
d) L'environnement est dynamique.
e) Le but local de chaque entité influe sur la sélection du comportement local à chaque étape.
f) Le but global du système est représenté par une fonction Fitness qui donne les mesures de la progression du calcul.

Parmi les applications de cette approche : la segmentation d'image

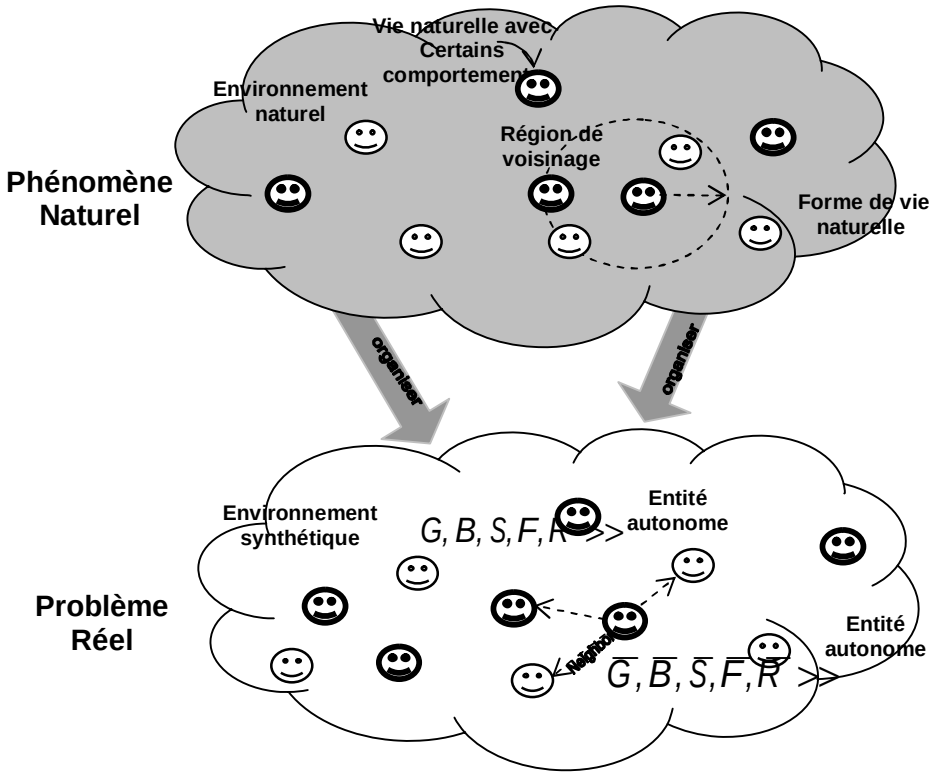


*Figure 3.34 :**A.O.C par fabrication***

3.4.2 A.O.C par prototypage (AOC-by-Prototyping) (Figure 3.35)

Lorsque les connaissances sur le mécanisme de fonctionnement d'un système complexe sont insuffisantes, donc il est difficile voir impossible de construire un modèle, on opte vers la création d'un modèle prototype. On utilise le processus essai erreur pour éliminer la différence entre le modèle prototype et son homologue naturel. Parmi les applications de cette approches la gestion des trafic automobile et la navigation sur internet.

A.O.C par prototypage est une application itérative de l'approche précédente avec l'ajout d'une phase d'ajustement des paramètres après chaque itération, qui dépend de la différence entre le comportement désiré et le comportement actuel du prototype.

Les caractéristiques de cette approche sont :

a) Les états, buts, comportements et règles de comportement peuvent être changés d'un prototype à l'autre.

b) La définition de l'environnement peut être aussi changée d'un prototype à l'autre.

c) Il existe une étape supplémentaire pour comparer le modèle synthétique avec son homologue naturel.

d) Un nouveau prototype est construit en adoptant les étapes a et b puis répéter tous le processus.

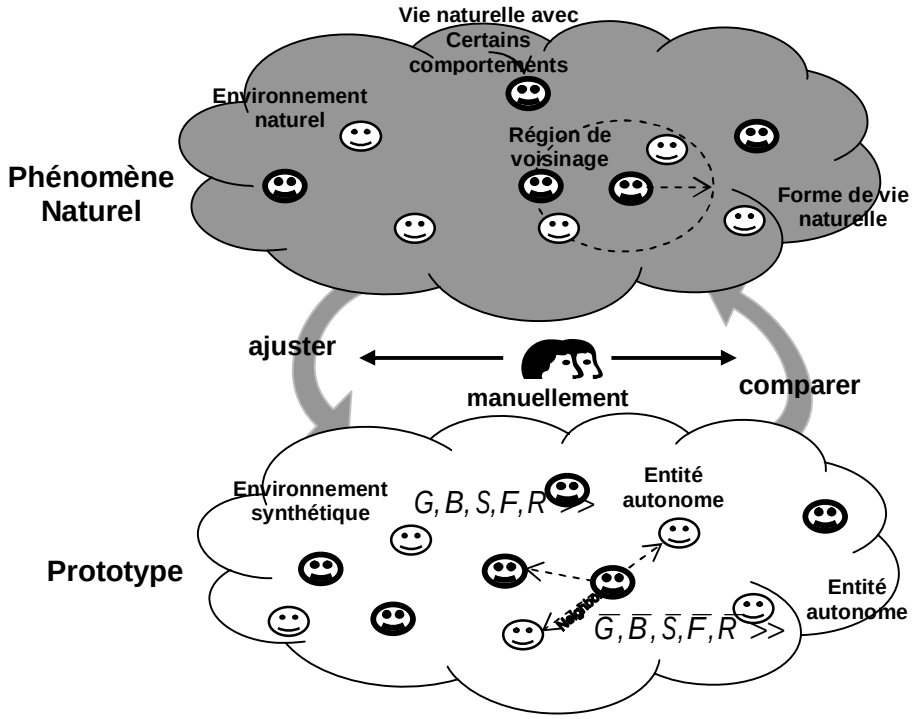


Fig 3.35 – ***A.O.C par prototyping***

3.4.3 A.O.C par auto-découverte (AOC by self-discovery) (figure 3.36)

Cette approche est basée sur l'aptitude d'un système de calcul de trouver son propre chemin afin de réaliser ce qu'un système basé sur l'approche A.O.C par prototypage peut faire. L'objectif final est d'atteindre une automatisation complète d'algorithme d'ajustement de ces propres paramètres. Dans les algorithmes évolutionnaires adaptatifs, les paramètres tels que le taux de mutation, l'opérateur de mutation, le taux de croisement, l'opérateur de croisement et la taille de population sont ajustés automatiquement. Un autre exemple de l'application de cette approche est l'introduction des algorithmes génétique pour évoluer un automate cellulaire pour une tâche de synchronisation. L'implémentation de cette approche est la même que celle de l'approche précédente à l'exception du processus essai erreur qui sera automatisé.

Cette approche a les mêmes caractéristiques de l'approche par prototypage avec l'ajout d'une autre caractéristique : les paramètres du systèmes sont auto adaptés selon les mesures de performance.

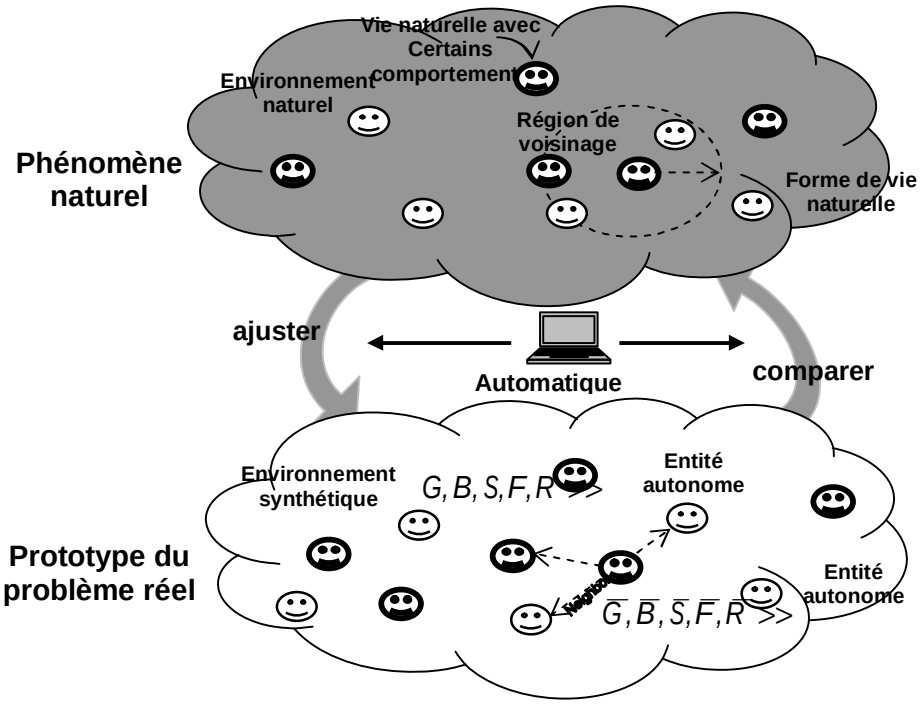


Figure 3.36 AOC par auto-détection

3.5 Un Framework de AOC

Les trois approches de AOC que nous venons de les présenter ont des structures et mécanismes similaires malgré la différence qui existe dans leur mode d'implémentation. Nous allons présenter quelques définitions formelles des éléments d'un système basé sur A.O.C à savoir les concepts : environnement, entité, interaction, et le processus d'auto organisation. Enfin nous formulons quelques exemples avec ce framework.

Avant d'aborder ces concepts, nous donnons une définition formelle d'un système AOC.

Définition 1 : Un système AOC est le triple <{e1, e2,…, e_N}, E, Φ> tel que :

{e1, …, e_N} : ensemble d'entités autonomes.

E : l'environnement où résident ces entités.

Φ : la fonction objectif.

3.5.1 Les éléments d'un système AOC

3.5.1.1 Environnement : C'est l'un des composantes principales d'un système AOC, il est le domaine d'existence des entités autonomes, et le moyen de communication indirecte entre ces entités

Définition 2 :

Un environnement **E** est caractérisé par l'ensemble :

$\mathcal{E}S = \{es_1, es_2, \ldots, es_{N\mathcal{E}S}\}$ avec $es_i \in D_{esi}$ représente un attribut statique ou dynamique.

A tout instant $\mathcal{E}S$ représente l'état actuel de l'environnement E.

Donc l'espace des états de E est donnée par $D_{\mathcal{E}S} = D_{es1} \times \ldots \times D_{esN\mathcal{E}S}$

Par exemple dans un processus de segmentation d'image, le niveau de gris d'un pixel représente l'attribut statique de l'environnement, par contre l'attribut dynamique c'est le marquage du pixel pour déterminer s'il appartient ou non à une région homogène.

3.5.1.2 Entités autonomes :

Les entités autonomes modifient leurs propres états, en interagissant avec l'environnement ou/et avec les autres entités. Chaque entité a un ensemble de comportements locaux et des règles de comportement définissant la façon d'agir ou de réagir en fonction des informations recueillies sur l'environnement et son voisinage.

Définition 3 : Une entité autonome **e** est le triple <**S**, **F**, **G**, **B**, **R**>

S : définit l'état actuel d'une entité e.

F : la fonction d'évaluation.

G : le but d'une entité.

B : ensemble de comportements.

R : règles de comportement.

Définition 4 : Le voisinage d'une entité e est un groupe d'entités noté :

$$L^e = \left\{ \mathbf{l}^e_1, \ldots, \mathbf{l}^e_{N_L} \right\}$$

Sachant que la relation (distance) entre chaque élément de voisinage $\mathbf{l}^e_i$ et l'entité e satisfait une certaine contrainte.

A tout moment chaque entité est dans un état selon ses règles de comportement, elle sélectionne et exécute une action pour arriver à un certain but, en interagissant avec son voisinage et/ou son environnement pour avoir l'information nécessaire.

Définition 5 : L'état d'une entité autonome **e** est caractérisé par un ensemble d'attributs dynamiques et statiques, et on note :

$S = \{S_1, S_2, \ldots\ldots, S_{NS}\}$

avec $S_i \in D_{si}$ et $D_S = D_{s1}$ x.....x D_{sNS} l'espace des états de l'entité **e**

Avant d'exécuter une règle de comportement, chaque entité doit vérifier sa condition qui dépend de son état interne, son voisinage et son environnement.

Définition 6 : l'estimation de conditions est réalisée par une entité en évaluant les fonctions suivantes :

a) l'état interne :

$$F : \hat{D}_s \rightarrow R$$

b) l'état de son environnement :

$$F : \hat{D}_{es} \rightarrow R$$

c) l'état de son voisinage :

$$F : \prod_{l_i^e \in L^e} \left(\hat{D}_{S_i^{l^e}} \right) \rightarrow R$$

d) l'état interne et celui de son environnement

$$F : \hat{D}_s \text{ x } \hat{D}_{es} \rightarrow R$$

e) l'état interne ainsi que l'état de son voisinage

$$F : \hat{D}_S \text{ x } \prod_{l_i^e \in L^e} \left(\hat{D}_{S_i^{l^e}} \right) \rightarrow R$$

Avec R domaine de valeur de F (par exemple l'ensemble des nombres réels);
$\hat{D}_s$ est le produit cartésien des éléments d'un sous ensemble de $\{\hat{D}_{si}\}$;
$\hat{D}_{es}$ est aussi un produit cartésien avec $\hat{D}_{es} \subseteq \hat{D}_s$
$\mathbf{l}_i^e$ dénote le i^{ème} voisinage de **e**

$s_i^{l^e}$ et $\hat{D}_{S_i^{l^e}}$ dénote l'état actuel et l'espace de la valeur d'une entité de voisinage $\mathbf{l}_i^e$

$\hat{D}_{S_i^{l^e}} \subseteq D_{S_i^{l^e}}$

$\prod$ est l'opérateur du produit cartésien

Définition 7 : Chaque entité est engagé à tout moment pour atteindre un ensemble de buts noté G, tel que : G = {g_1,……,g_{nG}}. Un but g_i est d'arriver à un état S' tel que la fonction d'évaluation F prend une certaine valeur prédéfinie a, i.e, gi = {S' / F(.) = α}

Définition 8 : L'ensemble de comportements locaux d'une entité **e** est B = {b1,b2,……,b_{NS}}. Un comportement b_i peut être :

a) Reproduction : $b_i : e \rightarrow e^m$ par duplication de l'entité elle-même m fois

b) Elimination : $b_i : e \rightarrow f$ par suppression totale de l'entité e de son environnement

c) Changement de son état interne : $b_i : \hat{D}_s \rightarrow \hat{D}_s$

d) Changement de son état interne ainsi que celui de son environnement : $b_i : \hat{D}_s \text{ x } \hat{D}_{es} \rightarrow \hat{D}_s \text{ x } \hat{D}_{es}$

e) Changement de son état interne ainsi que celui de son voisinage :

$$b_i : \hat{D}_S \text{ x } \prod_{l_i^e \in L^e} \left(\hat{D}_{S_i^{l^e}} \right) \rightarrow \hat{D}_S \text{ x } \prod_{l_i^e \in L^e} \left(\hat{D}_{S_i^{l^e}} \right)$$

Définition 9 : L'ensemble des règles de comportement **R** d'une entité e, tel que R = {r_1,r_2,……,r_{NR}}, chaque règle de comportement r_i sélectionne un ou plusieurs comportement pour les exécuter. Les règles de comportement sont classées en deux catégories :

- Règles basées sur l'évaluation : r_i : $Dom(F) \longrightarrow \{\hat{B}\}$

  avec *Dom(F)* dénote les valeur de domaine de la fonction d'évaluation F et $\{\hat{B}\}$ est un sous-ensemble de *B*

- règles basées sur les probabilités : $r_i : [0,1] \rightarrow \{\hat{B}\}$

  tel que à chaque sous-ensemble $\hat{B}$ est assigné une probabilité $p_{\hat{B}}$, qui sera fixée ou changée à tout instant.

3.5.1.3 Fonction objectif du système

Dans un système AOC, la fonction objectif Φ guide le système d'évoluer vers l'état global désiré, on distingue deux types de Φ:

a) fonction orientée état : $\Phi : \prod_{e_k \in \{e_i\}} \hat{D}_{S^{ek}} \rightarrow R^m$ avec

$\hat{D}_{S^{e_k}}$: est un sous-ensemble de l'espace des états d'une entité e_k

m : la dimension du domaine de valeur de Φ

Avec ce type de fonction, le système vise un certain état désiré.

b) fonction orientée processus : $\Phi : \left\{ \prod_{e_k \in \{e_i\}} \hat{D}_{S^{e_k}} \right\}^{\Pi} \rightarrow R^m$

Le système tente d'exhiber certaines caractéristiques désirées dans son processus évolutionnaire.

3.5.2 Les interaction dans un système AOC

Le comportement émergent d'un système AOC est le résultat des interactions locales, on a deux formes d'interaction :

a- interactions entre entités et leur environnement

b- interactions entre entités

Définition 10 : Les interactions entre une entité **e** et son environnement **E** est l'ensemble des traces $\{I_{eE}\}$ tel que I_{eE} prend l'une des deux formes :

$$I_{eE} : \hat{D}_{es} \rightarrow_{bi} \hat{D}_{es}$$

ou

$$I_{eE} : \hat{D}_{s} \text{ x } \hat{D}_{es} \rightarrow_{b_i} \hat{D}_{s} \text{ x } \hat{D}_{es}$$

avec "$\rightarrow_{bi}$" indique que I_{eE} est en fait le comportement b_i de l'entité e et ayant rapport avec le sous espace des états de l'environnement $\hat{D}_{es}$

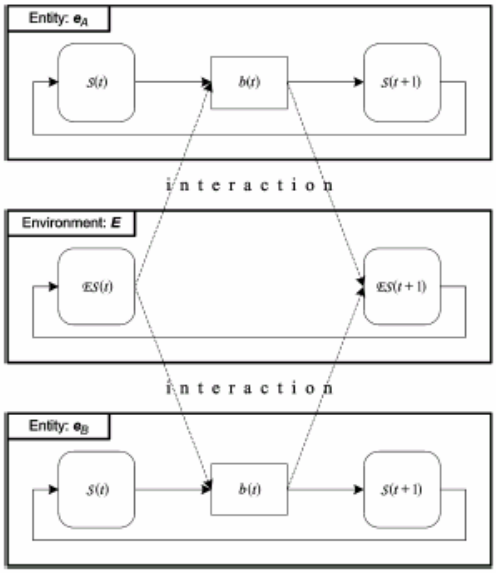


Figure 3.37 ***: Interaction entre deux entités e_A et e_B et leur environnement***

Définition 11 : les interactions directes entre les entités e_A et e_B est le tuple de l'ensemble des traces $\{I_{AB}, I_{BA}\}$ avec :

$$I_{AB} : \hat{D}_{S^A} \text{ x } \hat{D}_{S^B} \rightarrow b^A \ \hat{D}_{S^A} \text{ x } \hat{D}_{S^B}$$

ou

$$I_{BA} : \hat{D}_{S^B} \text{ x } \hat{D}_{S^A} \rightarrow b^B \ \hat{D}_{S^B} \text{ x } \hat{D}_{S^A}$$

avec $"\rightarrow b^A"$ et $"\rightarrow b^B"$ indiquent que I_{AB}, I_{BA} sont respectivement les comportements des entités e_A et e_B et ayant relation avec leur voisinage. $\hat{D}_{S^A}$ et $\hat{D}_{S^B}$ sont respectivement des sous ensembles des espaces des états des entités e_A et e_B

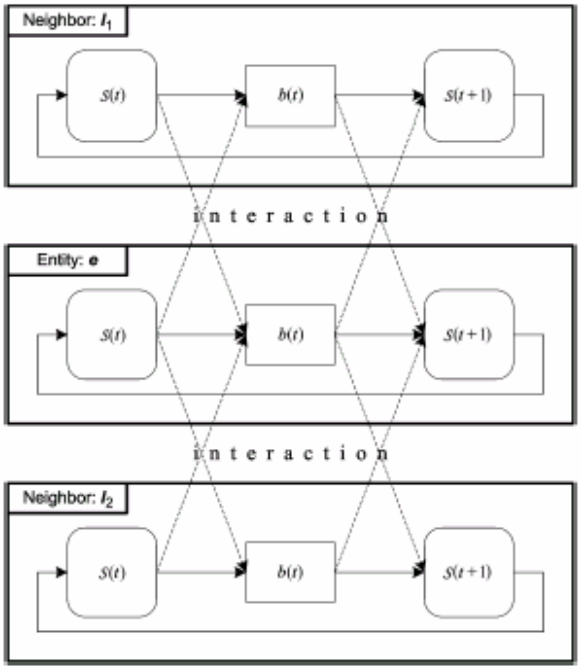


Figure 3.38 ***: interactions directes entre entité e et deux entités voisines l_1 et l_2***

3.5.3 Comportement du système

Le grand problème dans la modélisation d'un système AOC est de trouver la relation entre les comportements locaux et les règles de comportement des entités d'une part et le comportement complexe du système entier d'autre part.

A partir des comportements locaux des entités, ainsi que leurs interactions, un système AOC peut exhiber deux formes de comportement complexe : comportement émergent et comportement émergent intéressant.

Définition 12 : Un comportement émergent (Emergent behavior) : est le résultat des interactions entre entités individuelles, et qui n'est pas inhérent dans ces entités.

Définition 13 : Un comportement émergent intéressant (Emergent purposeful behavior) : est un comportement émergent dirigé par le but et qui ne peut pas être atteindre au niveau des entités individuelles.

3.5.4 L'auto-organisation dans un système AOC

Dans son parcours au but, une entité autonome sélectionne son comportement locale et exécute l'action associée en tenant compte des informations sur son état interne, sur son voisinage et sur l'environnement ceci est réalisé à travers des interactions directes avec les autres entités ou indirectes par la voie de son environnement, en partageant les informations.

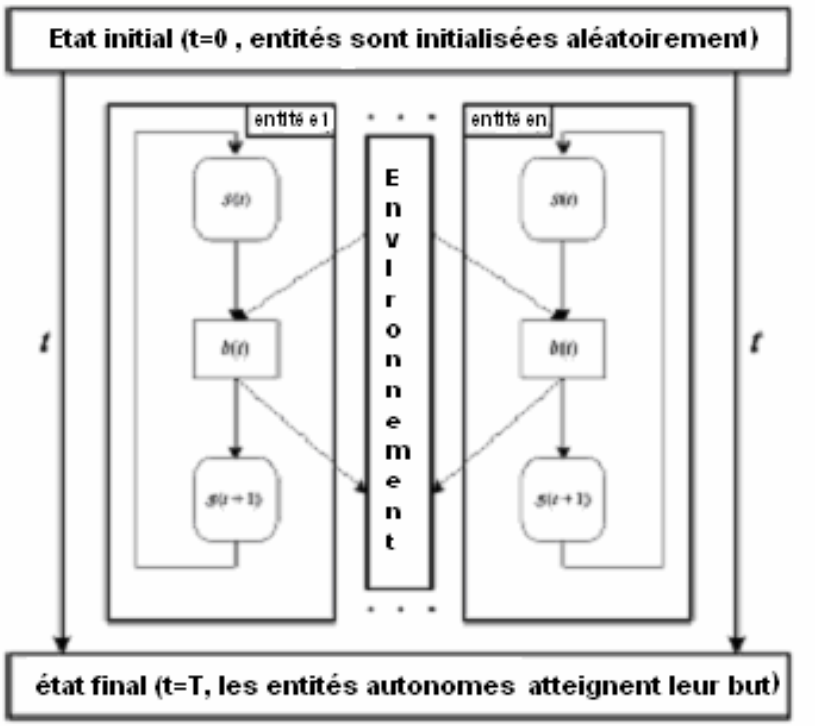


Figure 3.39: ***Processus d'auto-organisation dans un système AOC***

Définition 14 : Un processus d'auto-organisation des entités $\{e_i\}$ dans un système AOC est une séquence d'états de transition $\left\{\left\{S_t^{e_i}\right\}/\, t = 0,....,T\right\}$ soumise aux contraintes suivantes :

a) contrainte locale pour chaque entité :

$$\Pr(F(S^{e_i}{}_{t+1}) - F(S^{e_i}{}_t) > 0) \geq 0$$

avec $F(S^{e_i}{}_t)$ retourne la valeur d'évaluation de l'état de l'entité e_i à l'instant t et Pr(.) c'est la probabilité. Cela signifie que le nouvel état de l'entité e_i à l'instant t+1 est meilleur que celui à l'instant t

b) contrainte globale : $\Pr(\Delta\Phi_t > 0) \geq 0$

avec $\Delta\Phi_t = \Phi_t - \Phi_{t-1}$ qui représente le changement de la fonction objectif entre l'instant t et t+1

D'après ces deux contraintes, le système finalement atteint un état ou : $\Phi_T = \mathrm{Opt}(\Phi_t)$, où Opt(.) retourne la valeur optimale.

D'après cette définition, la contrainte locale indique que les entités autonomes évoluent durant chaque itération vers des états de plus en plus meilleurs et ceci d'une manière probabiliste, mais pas nécessairement toutes les entités. Par contre la contrainte globale indique que à chaque étape, tout le système AOC évolue vers un état meilleur selon sa fonction objectif d'une manière probabiliste

Définition 15 : AOC est un processus nécessitant les étapes suivantes :

Etape1 : Concevoir un système AOC < $\{e_1,e_2,\ldots,e_N\}$,E,Φ >, en particulier :

- concevoir les états internes de l'environnement $\mathcal{E}$S
- pour chaque entité autonome, concevoir ces états internes S, sa fonction d'évaluation F, ses buts G et ses règles de comportement R
- Concevoir la fonction objectif du système Φ.

Etape 2 : Déterminer approximativement la valeur désirée Φ^*de la fonction objectif du système Φ.

Etape 3 : Faire exécuter le système AOC conçu, puis obtenir une valeur final Φ'

Etape 4 : Définir une fonction à optimiser $\Psi = |\Phi' - \Phi^*|$ qui joue le rôle de guide dans un Système AOC

Etape 5 : Si Ψ n'est pas optimisée alors modifier ou ajuster les paramètres des entités et/ou d'environnement.

Etape 6 : répéter les étapes ci-dessus jusqu'au optimisation de Ψ

3.5.5 Exemples formulés par AOC

3.5.5.1 Segmentation d'une image à niveaux de gris [13]:

Dans cet exemple, les entités autonomes sont déployées sur une représentation 2-D d'une image à niveaux de gris, l'objectif est de localiser les régions homogènes. A partir d'une répartition aléatoire, les entités autonomes se diffusent pour explorer l'image entière, si une région homogène est détectée, elles reproduisent un nombre fixe d'entités qui continuent à leur tour la tâche prédéfinie. Finalement les régions satisfaisantes notre critère de segmentation seront étiquetées.

a) *Environnement* : L'image est considérée comme environnement où résident les entités autonomes, il est caractérisé par deux attributs :

- niveau de gris d'un pixel
- étiquetage d'un pixel pour indiquer s'il appartient ou non à une région homogène

L'environnement sera changé, en exploitant plus de pixels par les entités autonomes.

b) *Entités autonomes* :

1. L'état d'une entité est caractérisé par 4 attributs : position, age, durée de vie et l'activité La durée de vie est une constante prédéfinie, par contre les trois autres paramètres peuvent être changés dynamiquement.
2. Le but d'une entité est d'explorer l'image est de localiser certaine région homogène
3. La fonction d'évaluation permet de vérifier si le pixel occupé par une entité appartient ou non à une certaine région homogène, cela dépend des valeurs de contraste, de moyenne régionale et d'intensité de déviation du niveau de gris
4. Chaque entité a quatre comportements :
   - reproduction : par réplication de l'entité elle-même plusieurs fois ;
   - diffusion : par changement de position ;
   - étiquetage : en changeant l'état de l'environnement ;
   - disparition.
5. Règles de comportement : sélectionner à chaque étape un ensemble de comportements selon la fonction d'évaluation

c) *Les interactions* : une entité interagit avec son voisinage pour avoir l'information sur l'intensité du niveau de gris, ainsi elle peut étiqueter le pixel où elle réside. Dans ce cas, il n'existe pas d'interactions directes entre entités.

3.5.5.2 Système de recherche et extraction d'information sur le Web [12]:

Dans cet exemple, des entités de recherche d'information surfent sur le Web artificiel dans le but de rechercher et extraire les informations qui les intéressent. A chaque étape, ces entités vérifient si elles ont récoltées suffisamment d'information, dans ce cas elles doivent quitter le web, sinon elles continuent leur recherche jusqu'à satisfaction.

a) L'environnement : c'est l'espace artificiel Web, où résident les entités de recherche (les usagers). Dans cet environnement, toutes les informations sur les pages/sites Internet sont enregistrées.
b) L'entité : est un agent de recherche caractérisé par :
    1. Etat d'une entité caractérisé par les quatre attribut : position, motivation, récompense et intérêt. Ces attributs sont dynamiques, elles peuvent être modifiés à tout instant.
    2. Le but des entités est de trouver l'information voulue.
    3. La fonction d'évaluation : permet de juger si une entité a trouvé ou non assez d'information, ou pour exprimer la satisfaction ou non d'une entité.
    4. Le comportement : consiste à se déplacer d'une page/site web à l'autre en suivant plusieurs stratégies.
c) Les interactions : dans cet exemple, les interactions existent entre entités et leur environnement

3.5.5.3 Optimisation (Evolutionary Diffusion Optimisation EDO) [16][17]

Inspiré de la diffusion dans la nature, Jiming Liu et Tsui ont développé une méthode basée sur AOC appelée " Evolutionary Diffusion Optimisation EDO" pour résoudre un problème d'optimisation globale.

a) L'environnement : c'est l'espace de recherche spécifié par le problème d'optimisation. Pour chaque position, une solution possible du problème est enregistrée**.**
b) Une entité est formulée comme suit **:**
    1. L'état d'une entité : est représenté par les attributs : age, position, durée de vie, matrice de probabilité, activité,…
    2. Chaque entité explore l'environnement pour trouver une position permettant d'optimiser le problème donné.
    3. La fonction d'évaluation : permet d'évaluer la position actuelle d'une entité si elle meilleure ou non.

4. Les comportements : chaque entité a les comportement suivants : diffuser, reproduire, disparition et rajeunir

c) **Les interactions :** dans cet exemple on a deux types d'interaction : interactions directes entre entités par le biais du partage de la matrice de probabilité entre entité et ses des descendants, et interactions avec l'environnement en cherchant une solution possible.

Dans le paragraphe suivant nous allons exposer le principe de fonctionnement de cet algorithme

3.6 Algorithme E.D.O "Evolutionary Diffusion Optimisation"[16][17]

3.6.1 Introduction

L'utilisation des systèmes multi-agents dans la résolution distribuée des problèmes complexe représente une approche prometteuse. Par contre la majorité des systèmes multi agents nécessitent un certain degré de planification, de négociation et de coordination pour arriver au but. Cet algorithme représente une nouvelle approche multi agent inspirée de la diffusion dans la nature, l'ensemble des agents coopèrent pour réaliser une tâche d'optimisation d'une manière distribuée, dans l'absence d'une information complète sur l'espace de recherche, et ceci par le biais du partage et la mise à jour de leurs croyances. Aucun agent ne joue le rôle de planificateur ou de coordinateur central, ainsi qu'un minimum d'information est disponible. Les agents sont organisés en famille, telle que chaque famille est constituée d'un agent père et ses descendants. Chaque entité (agent) prend la décision de diffuser en se basant sur l'information partagée entre un agent père et ses descendants.

Cet algorithme fait parti de l'approche A.O.C par auto découverte "A.O.C by self discovery" qui a comme caractéristique principale l'ajustement automatique des paramètres (règles) lors de la recherche de la solution optimale d'un problème complexe donnée. Cet algorithme est présenté par Jiming liu et Kwok Ching Tsui [17]

Pour la conception de ce système multi agents les aspects suivants sont pris en considération :

- Un niveau de communication faible
- La taille de la population d'agents est variable. Par contre dans les algorithmes génétiques et les systèmes d'optimisations de colonies de fourmis gardent une taille fixe de population
- Les agents sont homogènes : ayant les mêmes ensembles de comportements.
- Les agents ont un comportement local

- Les agents sont rationnels à un certain degré, cela signifie que le choix de comportement à exécuter ne suit pas toujours les mêmes règles de comportement.
- Ils coopèrent par le biais du partage de leur croyance, au lieu de coopérer par négociation ou compétition.
- Les agents ne doivent pas forcément exister durant tous le processus de recherche.

3.6.2 Source d'inspiration

Le principe de l'algorithme E.D.O est inspiré de la diffusion dans la nature, par exemple dans la migration humaine deux facteurs importants influent :

1- Facteurs concernant les conditions défavorables de vie tels que le chômage et la pauvreté "Push factors".

2- Facteurs concernant les conditions favorables qui encouragent les gens à se déplacer tels que la disponibilité de l'emploi et le mode de vie amélioré "Pull factors".

On peut distinguer deux formes de migration :

- La migration par étape : est caractérisée par une série de mouvements locaux tel que le déplacement d'un village à une municipalité puis vers une ville.
- La migration enchaînée est caractérisée par un changement radical au-delà de la région locale, ce changement est réalisé par les gens qui viennent d'effectuer une migration.

La disponibilité de l'information est très importante, cela aide les gens à décider où et quand effectuer la migration. Ainsi les connaissances d'une catégorie de gens peuvent aider les autres à décider de se déplacer ou non. Autrement dit un mouvement bénéfique est réalisé par le biais du partage des connaissances et la chance de trouver un emplacement meilleure (solution) peut être augmentée en faisant des petits pas de migration.

En se basant sur ses observations nous allons décrire l'algorithme E.D.O pour une tâche d'optimisation d'une fonction.

3.6.3 Algorithme E.D.O

3.6.3.1 Problème à résoudre

Soit la fonction F(x) à optimiser avec $x = \{x_1, x_2, \ldots, x_n\}$ est un vecteur à n dimensions représentant les paramètres de la fonction.

La solution optimale est représentée par $F(x^*)$ tel que :

$$\text{"}\ \mathbf{x : F(x^*) < F(x)} \quad (1).$$

La recherche de x^* peut être vu comme minimisation de la fonction F , et en inversant le signe de la formule (1) ceci nous ramènera à maximiser la fonction F.

L'algorithme doit prendre en considération les contraintes suivantes :

- L'architecture de la fonction F (Landscape) est inconnue.
- La non linéarité de la relation entre les changements4 au niveau des variables de la fonction et leurs valeurs correspondantes.
- L'algorithme n'atteint pas directement la solution optimale mais en effectuant des changements de petits pas et d'une façon incrémentale
- Le maintien d'une population diverse pendant tout le processus de la recherche de solution optimale

3.6.3.2 Description d'agent dans EDO

Les agents dans E.D.O sont divisés en deux catégories :

1- Agents actifs : sont les agents qui sont à la recherche de la solution optimale, et ayant comme comportement : la diffusion et révision négative de leur croyance.

2- Agents inactifs : ce sont les agents ayant trouvés une solution meilleure que celle de leurs parents et qui n'est pas nécessairement la solution optimale. Ils ont comme comportement : la reproduction et la révision positive de leurs croyances.

Un agent est représenté par le tuple (P,S) tel que S est l'état d'un agent, et P est la matrice de probabilité qui représente ces croyances concernant la localisation de la solution optimale.

L'état de l'agent désigné par S est le triple (a, f, V) avec **a** est l'age de l'agent, **f** sa fitness et **V** son vecteur d'objet (voir détail ci-dessous)

3.6.3.2.1 Vecteur d'objet : le vecteur d'objet V peut être représenté comme suit :

$V = \{v_1, v_2, \ldots . v_n\}$, $v_i = [LB,UB]$, $\forall i$

LB et LU : sont les bornes supérieure et inférieure des valeurs prises par les variables de la fonction.

3.6.3.2.2 Matrice de probabilité : On associe à chaque agent une matrice de probabilité P de dimension n x m avec **n** c'est le nombre de variables de la fonction et **m** le nombre des pas possibles de déplacement. Elle représente les croyances d'un agent dans le système multi agent proposé.

Δ : est un paramètre global indiquant la taille du pas, il représente le changement de tous les variable de la fonction

Le produit de Δ et le nombre de pas représente les modifications finales effectuées au vecteur V.

Soit : $P = \{ P_1, P_2, \ldots\ldots P_n \}$

$P_i = \{P_{i,-y}, \ldots.., P_{i,0}, \ldots\ldots P_{i,y}\}$, $0 \leq p_{i,j} \leq 1$

avec $y = (m-1)/2$ et $\sum_{j=-y}^{y} P_{i,j} = 1 \quad \forall i$

Initialement tous les éléments de la matrice P prennent la valeur 1/m , cela signifie que les agents peuvent effectuer n'importe quel mouvement possible.

La matrice P est mise à jour d'une façon continue, c'est le résultat d'un comportement local d'un agent (Feedback). A tout instant les tous les agents transmettent l'information à leurs parents s'ils ont effectué un bon déplacement vers une meilleure localisation ou l'inverse. Cette information permet à un agent père de mettre à jour la matrice P partagée avec ces descendants.
Les agents pères et leurs descendants partagent la même matrice de probabilité et lorsqu'un agent devient père une matrice de probabilité sera créée pour lui dont le contenu est une copie intégrale de la matrice de l'agent père. Le partage l'information représentée par la matrice de probabilité entre parents et descendants leurs permet d'apprendre les meilleurs stratégies.

3.6.3.2.3 La fonction de performance Fitness :

La fonction Fitness évalue le degré de succès d'un agent dans le processus de la recherche de la solution optimale. Pour simplifier, fitness prend la valeur objective de la fonction dans l'algorithme E.D.O s'il s'agit d'une tâche de minimisation.

3.6.3.2.4 Le comportement local :

Initialement tous les agents sont considérés actifs, ils ont deux comportements : diffusion et feedback, si un agent est diffusé à un emplacement avec une valeur de fitness supérieure à celle de son père il va reproduire un nombre d'agents puis il devient inactif.

3.6.3.2.5 Diffusion

A chaque instant, les agents diffusent vers de nouveaux emplacements selon la distribution de la probabilité donnée par la matrice de probabilité P.
On distingue deux stratégies :

- Mouvement rationnel "Rational move" : L'agent modifie sont vecteur d'objets

En choisissant le nombre de pas à prendre selon la matrice de probabilité P :

$vi = vi + \delta vi \,.\, \Delta \quad \forall i$

- Mouvement aléatoire "Random Walk" : Si après plusieurs itérations, l'agent n'arrive pas à localiser un emplacement meilleure que celui de son parent, alors il décide d'effectuer un mouvement aléatoire.

  La probabilité d'effectuer un mouvement aléatoire P_{rw} est donnée par la formule :

$$P_{rw} = \exp[-\frac{q-a}{a}]$$

a : est un paramètre scalaire indiquant le degré d'exécution d'un mouvement aléatoire.

a : age d'un agent

q : durée de vie maximale d'un agent

3.6.3.2.6 Reproduction :

A la fin de chaque itération, la fonction fitness de chaque agent actif est comparée à celle de son parent qui est devenu temporairement inactif, tous les agents ayant une fitness supérieure sont autorisés à reproduire. L'entité reproduite est une duplication de l'entité reproductrice, les nouvelles entités se dirigent vers nouveaux emplacements par un mouvement rationnel.

Le nombre d'entités à reproduire q_x (quota) dépend de la valeur de la fonction fitness et deux paramètres du système à savoir :

Ω : le nombre maximum d'entités à reproduire

Π : la taille maximale de la population d'agents

$$q_x = \begin{cases} \Omega, si & \frac{f_x}{f} \le 0.25 \\ \Omega - 1, si & 0.25 \le \frac{f_x}{f} \le 0.5 \\ \Omega - 2, otherwise & \end{cases}$$

f : la moyenne de fitness

x : l'agent

3.6.3.2.7 Disparition : à la fin de chaque itération, l'age de chaque agent actif est incrémenté d'une unité. Si l'agent atteint sa durée de vie, il sera éliminé du système, ce qui nous permet une bonne exploitation de l'espace de recherche, mais ça nous n'empêche pas à garder les agents

ayant une bonne performance et d'éliminer prématurément les agents ayant une mauvaise performance.

Donc on peut avoir les cas d'exception suivants :

- Extension de la durée de vie d'un agent si sa fitness est au dessus de la moyenne.
- Elimination d'agent ayant une fitness inférieure à celle de l'agent père avec un certain pourcentage (80% par exemple)
- Rajeunissement : un agent inactif est autorisé à reproduire si tous ses descendants sont éliminés du système et que sa fitness est dessus de la moyenne.

3.6.3.2.8 Feedback

A tout instant les agents transmettent l'information à leurs parents s'ils ont effectué un bon mouvement pour un emplacement meilleur ou un mauvais mouvement, ces informations permettent aux agents parents de mettre à jour la matrice de probabilité partagée avec leurs descendants. Ce mécanisme est appelé feedback on peut avoir 2 types :

Feedback positif : correspond à un déplacement bénéfique et définie par un gain dans la fonction fitness. Dans le but de guider positivement les mouvements futurs des agents descendants, la probabilité correspondante au changement effectué dans le vecteur d'objet dans la matrice de probabilité est donnée par la règle :

$$p'_{i,j} = \frac{\left(p_{i,j} + b\right)}{\left(1+ b\right)}$$

$p_{i,j}$ **:** La probabilité correspondante à la variable d'ordre i et le pas j

b : Taux d'apprentissage

Feedback négatif : pour conduire les agents à éviter les zones non optimales dans l'espace de recherche, la matrice de probabilité de l'agent parent est mise à jour après chaque mouvement sans succès suivant la formule :

$$p'_{i,j} = p_{i,j} \,.\left(1-b\right)$$

3.6.3.2.8 Paramètres généraux :

- Δ : la taille d'un pas lors d'un mouvement rationnel.
- Ω : le nombre maximum d'agents reproduits
- Π : Le nombre maximum d'agents actifs et inactifs pendant une session dans l'algorithme E.D.O
- q : La durée de vie exprimée en nombre d'itérations attribués à chaque agent.

3.7 Le pseudo code d l'algorithme E.D.O

Début
Tant que *(nombre d'agents > 0)* ***et*** *(nombre d'itérations < limite)* ***faire***
Evaluer l'agent
Si *(sa performance est meilleure que celle de son parent)* ***alors***
Feedback positif
Reproduire
Agent parent devient inactif
Sinon
Feedback négatif
Diffuser
vieillir
Fin-si
Fin*-tant que*
Fin

Procedur*e Reproduire*
Début
Quota ⟵ f(fitness)
Creation d'une nouvelle matrice de probabilité
Pour *chaque agent reproduit* ***faire***
Copier tous les variable
Pointer vers la nouvelle matrice de probabilité
Diffuser
Fin-pour
Fin

Procedure *vieillir*
Début
age ⟵ age + 1
Si *(fitness< fitness-parent*seuil)* ***ou*** *(age> duree-de-vie et fitness < moyenne)* ***alors***
Supprimer l'agent
Fin-si
Fin

Procedure *Diffuser*
Début
Si *random() > P_{rw}* ***alors***
Pas aléatoire
Sinon
Pour *chaque variable* ***faire***
Selectionner la taille du pas et la direction
Fin-pour
Fin

3.8 Algorithmes évolutionnaires et Algorithme EDO

La différence entre l'approche E.A et l'algorithme E.D.O, est résumée en sept point :

- La taille de la population d'agents est fixe dans les algorithmes évolutionnaires, par contre, elle est variable dans E.D.O ce qui permet aux entités ayant de faibles chances de trouver la solution optimale de survivre.
- L'opération de sélection dans les algorithmes évolutionnaires est une opération basée sur le tri de la valeur de fitness de tous les agents, par contre dans E.D.O , elle est locale et ne concerne que le voisinage d'un agent ce qui rend l'opération mois coûteuse et d'avoir un parallélisme meilleur.
- La matrice de probabilité dans E.D.O permet aux agents de capturer la tendance de fitness, cette information est locale, partagée entre l'agent parent et ses descendants, par contre dans les algorithmes évolutionnaires cette tendance ne peut être observée que par la totalité des agents.
- Dans les algorithmes évolutionnaires , la chance de reproduire ou de rester démunie en fonction de la valeur de fitness d'un agent, par contre dans E.D.O, elle dépend seulement de son voisinage locale, ce qui engendre une diversité dans la population. On peut dire que la reproduction est réalisée par compétition locale.
- Le pas aléatoire joue un rôle puissant dans EDO, par contre dans les algorithmes évolutionnaires c'est un outil pour échapper de la solution non optimale. C’est-à-dire que le mouvement aléatoire dans EDO est introduit comme opérateur de diffusion
- On dispose dans EDO d'un minimum d'information globale
- Adaptation du pas de recherche pendant le processus d'optimisation

3.9 Tableau comparatif entre OOP, AOP et AOC [41]

	Programmation orienté objet (OOP)	**Programmation orientée agent (AOP)**	**A.O.C**
Elément de base	**Objet**	**Agent**	**Entité autonome et environnement**
Caractéristiques de l'élément de base	**Variable ou fonction**	**Croyances, décisions, aptitudes et obligations**	**Etat, fonction d'évaluation, buts, comportement de base et les règles de comportement**
Interaction	**Héritage et envoi de messages entre agents**	**Messages entre agents, incluant les requêtes informer, demande, offre, promettre et décliner**	**(1) interaction entre entités et leur environnement** **(2) interaction directe ou indirecte entre entités**
Calcul	**Envoi de messages et exécution de méthodes**	**Envoi de messages et exécution de méthodes**	**(1) Agrégation de comportement et interaction** **(2) auto-organisation des entités autonomes**
Convient à	**(1) Modélisation des systèmes** **(2) Calcul basé sur réutilisation de code**	**(1) Développement de systèmes distribués** **(2)Résolution des problèmes distribués**	**(1)Résolution des problèmes difficiles** **(2) caractérisation des comportements des systèmes complexes**

Chapitre IV

A.O.C & SEGMENTATION D'IMAGES

4. A.O.C & Segmentation d'images

4.1 Problématique

On dispose d'une population d'agents réactifs, c'est-à-dire ne fonctionnant que sur le principe de perception/action, situés dans un environnement complexe et dynamique, ayant comme caractéristiques principales d'être autonomes et présentent des capacités d'adaptation et d'auto-organisation. Notre travail consiste à faire apprendre aux agents réactifs une tâche bien précise qui est le résultat des interactions soit entre agents eux-mêmes, soit entre agents et leur environnement, et ceci dans l'absence d'un agent central de contrôle et de coordination. Cette tâche consiste à localiser les régions homogènes dans une image numérique à niveaux de gris.
Ce travail est réalisé à travers trois étapes essentielles :

- **1ère étape :** est basée sur l'approche A.O.C par fabrication. Cette étape consiste à faire apprendre à un ensemble d'entités autonomes une tâche de segmentation d'image numérique à niveau de gris en s'inspirant du phénomène physique de la diffusion dans la nature. L'apprentissage est nécessairement incrémental, c'est-à-dire que le système doit apprendre au fur et à mesure de son expérience.

- **2ème étape :** Cette étape est consacrée à l'optimisation des paramètres de segmentation à savoir : le nombre d'agents initialement introduits, le nombre de progéniture durant un processus de reproduction, la durés de vie d'un agent et le rayon de diffusion.

- **3ème étape :** Est basée sur l'approche A.O.C par auto-découverte. Cette étape consiste à ajuster automatiquement les règles de comportement en fonction d'évolution de l'environnement.

4.2 La segmentation d'images

Une étape primordiale pour l'interprétation d'une image est la segmentation. La segmentation d'une image consiste à isoler et classer les pixels d'une image en catégories. Une multitude de méthodes de segmentation existe, mais aucune n'est absolue et optimale. En effet, la complexité des images à traiter (texture, détails fins, ...) rend difficile la conception d'une méthode générale. De plus, il est difficile de dire si une méthode de segmentation est meilleure qu'une autre dans un contexte donné. Pour évaluer ces méthodes, des critères d'évaluation ont été

développés. Ils peuvent utiliser ou non une information a priori sur les données à segmenter. Cependant, rien ne nous garantit la fiabilité des critères dans une situation donnée.

Notre objectif est l'application d'un nouvel algorithme basé sur le nouveau paradigme A.O.C et l'évaluation de divers résultats de segmentation, avec détermination des meilleurs paramètres de cet algorithme de segmentation d'où la nécessité d'utiliser une approche évolutionnaire.

Habituellement, l'efficacité d'un nouvel algorithme est illustrée par la présentation de quelques résultats de segmentation, ce qui n'autorise que des conclusions subjectives et qualitatives sur les performances de cet algorithme. Un critère d'évaluation de la qualité d'un résultat de segmentation peut alors présenter de nombreux intérêts. Par exemple, il peut permettre de faciliter le choix du paramétrage d'une méthode de segmentation, ou encore autoriser la comparaison de différentes méthodes de segmentation, voire même, de définir de nouvelles méthodes de segmentation (par optimisation du critère). Il pourrait aussi permettre de fusionner des résultats de segmentation en prenant en compte les valeurs des critères comme indice de confiance. Cependant, il n'existe pas à l'heure actuelle de critère absolu et générique pour effectuer cette tâche d'évaluation.

Nous considérons qu'il existe deux grandes classes de critères d'évaluation :

- l'approche non supervisée qui rassemble les critères d'évaluation ne nécessitant aucune connaissance sur les résultats de segmentation à évaluer. Leur principe consiste à estimer la qualité d'un résultat de segmentation à partir de statistiques calculées sur chaque région, contour ou texture détectés,
- l'approche supervisée qui regroupe les critères qui évaluent la qualité d'un résultat de segmentation en exploitant des connaissances a priori. Ces connaissances consistent le plus souvent en une comparaison par rapport à une image segmentée de référence.

Il existe deux principales approches en segmentation : l'approche frontière et l'approche région.

- Approches frontières : consiste à la détermination du contour. Un contour est une frontière entre deux différents milieux (2 couleurs, 2 niveaux de gris,...).
- Approches régions : dans ce second type d'approche, on ne suppose plus comme dans la première approche que l'on peut trouver des contours autour des différents objets que l'on veut mettre en évidence. On cherche plutôt à trouver dans quelle région de l'image ces objets se trouvent.

4.3 Première étape

Notre travail est un complément du travail réalisé par Jiming liu [13], notre contribution se traduit par l'implémentation du système réalisé sur une plate forme multi-agent (langage Netlogo) ainsi l'introduction de quelque modifications qui servent à optimiser les performances du système réalisé.

4.3.1 Principe

Étant donné, une image à niveaux de gris S, qu'on peut la représenter par un tableau à deux dimensions de taille U x V, dont chaque élément est un pixel. Les pixels appartiennent aux différends segments homogènes. Les caractéristiques d'un segment homogène est définie par les 3 critères : un critère de contraste, la moyenne régionale et l'écart type régional.

Une population d'entités autonomes est déployée sur l'image tel que chaque entité est capable d'estimer si un pixel appartient ou non à une région homogène prédéfinie en interagissant avec son environnement représenté par son voisinage. Lorsqu'une région non homogène est rencontrée, les entités autonomes diffusent vers un autre pixel dans le but de trouver une région homogène en suivant une certaine direction. Lorsque une entité autonome rencontre une région homogène elle va reproduire un certain nombre de progénitures et le pixel sur lequel elle réside sera étiqueté. Les entités reproduites auront la même tâche que celle de l'entité reproductive, c'est-à-dire la localisation des pixels appartenant à la même région. On remarque que chaque entité perçoit localement son environnement (voisinage) puis elle choisit et exécute le comportement approprié.

Les entités autonomes peuvent être diffusés d'un pixel à l'autre, se reproduire ,étiqueter ou simplement se disparaître entièrement de l'image (figure 4.40).

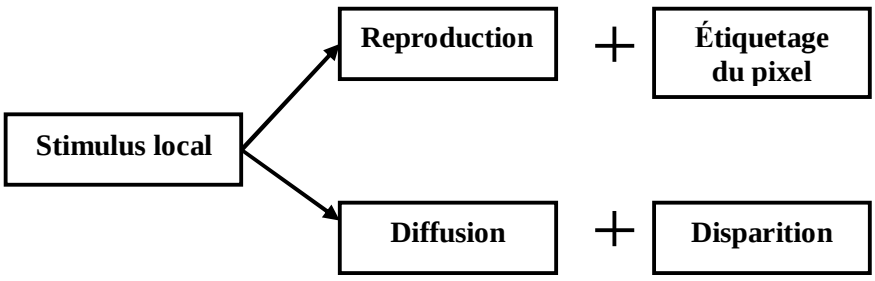


Figure 4.40 : ***Comportements d'une entité autonome***

Chaque entité a une durée de vie limitée, ce qui permet d'avoir une population variée et bien répartie ainsi une exploration meilleure et optimale de l'espace de recherche. En attribuant à chaque entité un état correspondant.

Notre système doit faire apprendre aux entités autonomes de choisir les meilleures directions pour atteindre une région homogène lors d'un comportement de diffusion. Un vecteur de diffusion est associé à chaque entité représentant la probabilité de succès de trouver une région homogène si une direction donnée est choisie.

Pour avoir une meilleure performance, on va exploiter le parallélisme de notre système en divisant les entités actives en famille d'entités. Ce qui rend la tâche d'apprentissage distribuée.

4.3.2 Agent de segmentation

La segmentation d'une image consiste à trouver les régions homogènes.

Deux classes d'agent peuvent être utilisés : Agents contour et agents région. Chaque classe va étiqueter une région homogène spécifique.

4.3.3 Stimulus local

A tout instant, chaque entité autonome (Agent) située sur un pixel quelconque de l'image, perçoit son environnement représenté par l'ensemble des pixels de son voisinage en évaluant les trois critères à savoir : le critère de contraste, la moyenne régionale et l'écart type régional. Cette évaluation permet à un agent de choisir une règle de comportement et d'exécuter le comportement correspondant.

Mathématiquement, une région homogène dans un emplacement de coordonnée (i, j) de l'image S peut être spécifiée en utilisant les critères suivants :

- Le critère de contraste $G_{(i,j)\text{-région}} \in (\eta_1, \eta_2)$.
- La moyenne régionale $Mean_{(i,j)\text{-région}} \in (M_1, M_2)$.
- L'écart type régional $Std_{(i,j)\text{-région}} \in (D_1, D_2)$.

Avec η_1, η_2 , M_1, M_2 , D_1 et D_2 des constantes prédéfinies.

4.3.4 La région de voisinage

La région de voisinage d'un agent à l'emplacement (i, j) est une région circulaire dont le centre est situé à l'emplacement (i, j) et le rayon est $R_{(i,j)\text{-région}}$. Les pixels situant à l'intérieur de cette région sont appelés le voisinage d'un agent (Figure 4.41).

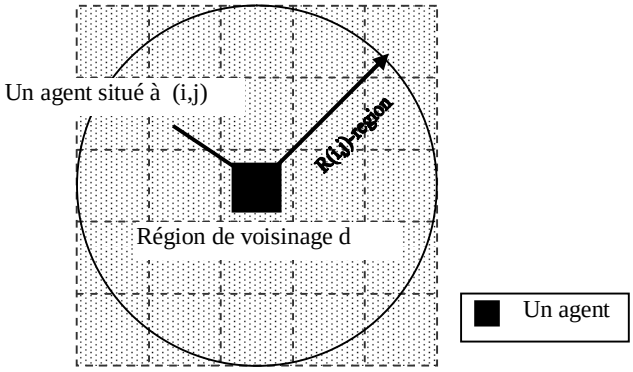


Figure 4.41 ***Voisinage d'un agent***

4.3.5 Critère de segmentation

* Le critère de contraste permet d'avoir le nombre de pixels dans une région de voisinage d'un agent dont la valeur d'intensité du niveau de gris est proche à celle du pixel ou réside l'agent. Le critère de contraste est définit comme suit :

$$G_{(i,j)-region} = \sum_{\|(i,j) - (k,l)\| \le R_{(i,j)\text{ - region}}} p(i,j,k,l)$$

$$p(i,j,k,l) = \begin{cases} 1 \; if \left\| I(i,j) - I(k,l) \right\| \le d, \\ 0 \; \text{sinon} \end{cases}$$

R : Rayon de la région du voisinage centrée à l'emplacement (i, j).

I(i, j) : Intensité du niveau de gris à l'emplacement (i, j).

δ : Constante prédéfinie représentant le seuil de contraste.

* L'expression de la moyenne régionale est donnée par :

$$mean_{(i,j)-region} = \frac{1}{N} \sum_{\|(i,j) - (k,l)\| \le R_{(i,j)\text{ - region}}} I(k,l)$$

Avec N est le nombre de pixels appartenant à une région de voisinage centré à l'emplacement (i, j).

* L'écart type régional est donné par l'expression.

$$std_{(i,j)-region} = \sqrt{\frac{1}{N} \sum_{\|(i,j)-(k,l)\| \leq R_{(i,j)-region}} \left(I(k,l) - mean_{(i,j)-region}\right)^2}$$

Après avoir détaillé, les différentes notions, tel que la région de voisinage, le stimulus local et le critère d'évaluation d'une région homogène nous allons construire notre système basé sur l'approche AOC par fabrication et ayant comme but essentiel la recherche et l'étiquetage des régions homogènes dans une image numérique à niveau de gris.

4.3.6 Modèle AOC

En se basant sur l'approche AOC par fabrication, la modélisation de notre système nécessite 3 phases :

4.3.6.1 Phase 1 (Identification du système naturel)

L'objectif principal de cette phase est de trouver une analogie entre le système complexe à résoudre et le système naturel approprié dans notre cas, on s'est inspiré du phénomène de la diffusion dans la nature. Cette phase nécessite la réalisation de deux tâches :

- Identification du comportement désiré du système.
- Identification des paramètres du système.

4.3.6.1.1 Identification du comportement désiré du système

La segmentation d'image permet d'identifier les régions homogènes dans une image. Un ensemble d'entités autonomes est déployé sur une représentation à deux dimensions d'une image à niveau de gris. Ces entités sont homogènes, dotées de capteurs permettant de vérifier si un pixel appartient ou non à une région homogène donnée. L'homogénéité est déterminée en fonction des trois critères :

- Critère de contraste.
- Moyenne régionale.
- Écart-type régional (déviation du niveau de gris).

La composition de la population change à tout moment, soit par un processus de naissance (amplification du comportement désiré) soit par le processus de la mort (élimination des comportements indésirables).

Si une entité autonome s'est retrouvée sur un pixel appartenant à la région homogène recherchée, elle reproduit un nombre d'entités identiques qui seront diffusées selon une certaine direction et qui seront eux-mêmes mobilisées dans le processus de détermination des régions homogènes. Par contre si une entité autonome s'est retrouvée sur un pixel n'appartenant pas à une région homogène, elle se diffuse vers d'autres emplacements. Les directions de diffusion et de reproduction sont définies respectivement par leurs vecteurs de comportement qui contiennent des valeurs entre 0 et 1 correspondants aux directions possibles.

Les entités autonomes interagissent localement et ceci dans l'absence d'une information globale qui permet de contrôler la sélection d'un comportement.

4.3.6.1.2 Identification des paramètres du système

Dans cette étape on va fixer le nombre d'entités qui seront déployées initialement sur l'image, leur durée de vie, ainsi le nombre d'entités reproduites pendant un processus de reproduction.

4.3.6.2 Phase 2 (Construction du système artificiel)

L'objectif de cette phase est la modélisation des entités autonomes de l'environnement.

4.3.6.2.1 Modélisation de l'environnement

Consiste à identifier les caractéristiques de l'environnement ainsi sa représentation.

La représentation 2-D de l'image à niveau de gris est considérée comme environnement. Il est caractérisé par deux attributs :

- Niveau de gris de chaque pixel.
- Étiquetage d'un pixel s'il appartient à une région homogène.

L'environnement est dynamique, il peut être changé d'un moment à l'autre.

4.3.6.2.2 Modélisation des entités autonomes

Définir le voisinage

Voir définition 4.3.4

Caractéristiques des entités :

Chaque entité déployée sur la représentation 2-D de l'image à segmenter est caractérisée par :

Etat : défini par :

- Position : qui indique les coordonnées (i,j) du pixel occupé.
- Age : Initialement il est à 1 et sera incrémenté d'une unité après chaque exécution d'un comportement. Si l'age dépasse la durée de vie prédéfinie, l'agent devient inactif.

- Durée de vie : qui est définie par le système pour tous les agents.
- Activité : si l'agent est à la recherche d'une région homogène et que son âge et inférieur à la durée de vie, il est actif. Par contre s'il rencontre une région homogène, il devient inactif.
- Vecteur de comportement.

On remarque que la position, l'activité, l'age et le vecteur de comportement sont des paramètres dynamiques.

But : Chaque entité est engagée dans le processus de recherche de pixels appartenant à une région homogène.

Définition. : Recherche optimale de régions homogènes :

Soit M dénote le nombre total de pixels appartenant à la région homogène de l'image I. si le nombre total de pixels détectés et étiquetés par les entités est égale à M on dit qu'une recherche optimale est aboutie.

Fonction d'évaluation

Permet l'évaluation d'un pixel s'il appartient à une région homogène, cela dépend des 3 critères (contraste, Moyenne régional, écart type régional).

4.3.6.2.3 Les comportements

La reproduction (Breeding)

Le comportement de reproduction est formellement donné par l'expression :

BR : Si stimulus local dans l'emplacement (i, j) vérifie les critères d'homogénéité.

$$\text{Alors : } a_{(i,j)}^{(g)} \Rightarrow \left\{ a_{u(\mathrm{w},\mathrm{l})} \middle| \upsilon = 1,2,...\mathrm{m};\ \mathrm{w} \in \Omega;\ \mathrm{l} \le \mathrm{K} \right\}^{(g+1)}$$

Avec : g, g+1 : Les générations de l'agent.

(i, j) : Les coordonnées de l'emplacement de l'agent.

α : un agent actif.

ν : $\nu^{ème}$ agent reproduit.

m : le nombre total des agents reproduits.

ω : La direction prise par un agent reproduit.

Ω : L'ensemble des directions possibles.

l : la distance entre l'agent reproducteur et l'agent reproduit.

K : le rayon de la région de reproduction.

Notre proposition consiste à donner la possibilité aux entités reproduites de choisir aléatoirement une direction de diffusion, ce qui permet d'avoir plus de chance pour la détection des régions homogènes (Figure 4.42).

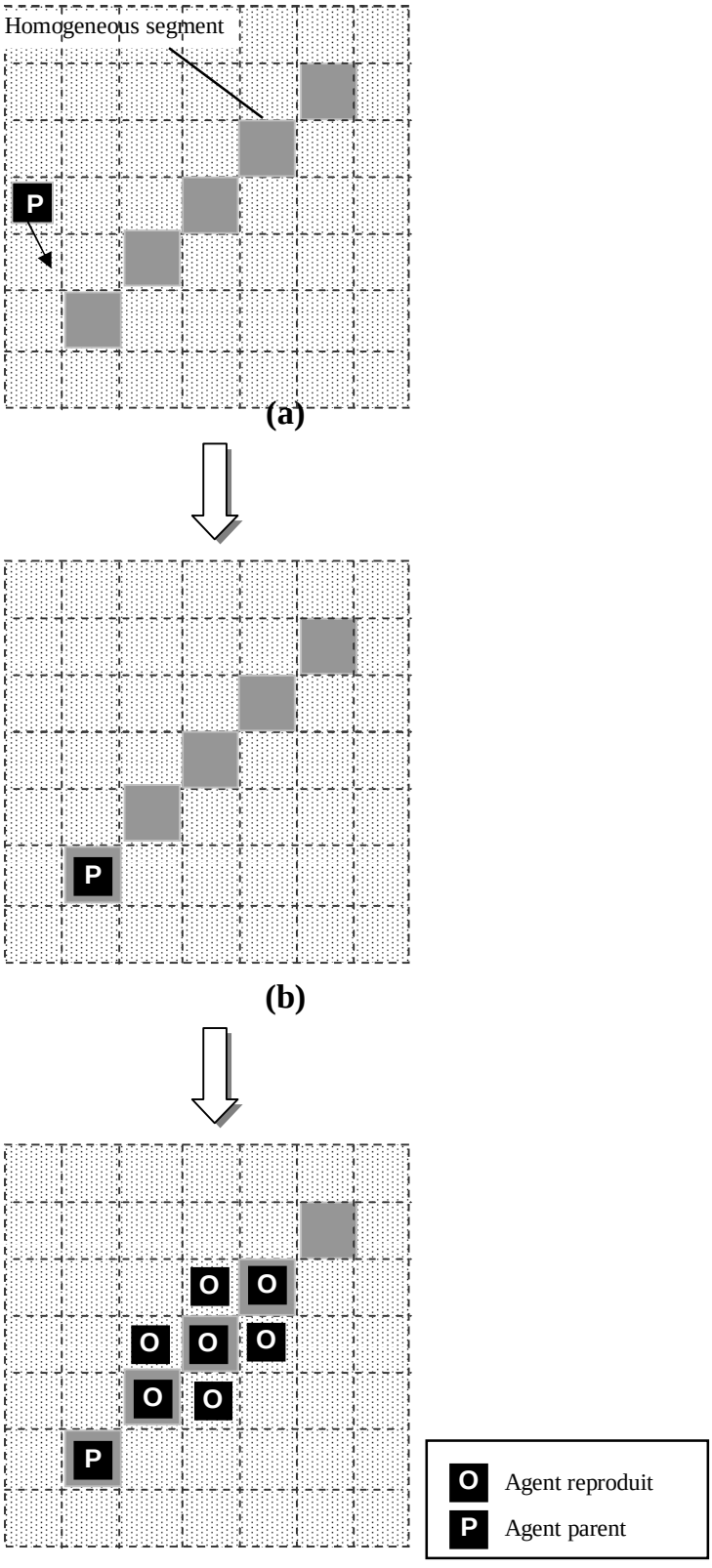


Figure 4.42 : ***Comportement de reproduction***

La diffusion (Breeding)

Le comportement de diffusion est exprimé formellement par l'expression :

Search : Si Stimules local à l'emplacement (i,j) ne vérifie par les critères d'homogénéité.

$$\underline{\text{Alors}} : a_{(i,j)}^{(t)} \Rightarrow a_{(q,d)}^{(t+1)},\ q \in \Theta,\ d \leq k$$

t, t+1 : Pas du temps.

(i, j) : Coordonnées de l'emplacement actuel de l'agent.

α : L'agent actif.

θ : Direction de diffusion.

Θ : L'ensemble des directions de diffusion possibles.

d : distance de diffusion.

K : rayon de la région de diffusion.

Le choix de la direction de diffusion est probabiliste. Initialement les agents (entités autonomes) prennent des directions aléatoires du fait que leurs vecteurs de comportements ont les mêmes valeurs de probabilité (par exemple 1/8 dans le cas d'un voisinage de 8).(Figure 4.43) Après chaque itération, le vecteur de comportement est mis à jour en bénéficiant de l'expérience acquise par les autres agents qui ont pu localiser une région homogène.

Le vecteur de comportement est donné par l'expression.

$$Q_q(\Theta) \stackrel{\Delta}{=} [q_1, q_2, ..., q_q, ..., q_N]$$

tel que : $q_q \in [0,1]$ et $\sum_{q=1}^{N} q_q = 1$, N : nombre de direction possible.

Étiquetage

Lorsqu'un agent rencontre un pixel appartenant à une région homogène, ce pixel sera étiqueté et l'agent devient inactif. Formellement l'étiquetage est donné par l'expression.

LABEL : Si stimulus local dans l'emplacement (i,j) vérifie les critères d'homogénéité.

$$\underline{\text{Alors}} : a_{(i,j)}^{(g)} \Rightarrow b_{(i,j)}^{(g+1)}$$

g, g+1 : génération de l'agent.

(i,j) : Les coordonnées de l'emplacement de l'agent.

α : L'agent actif situé à l'emplacement de coordonnée (i,j).

β : pixel étiqueté par l'agent situe à l'emplacement de coordonnée (i,j).

Le nombre de pixels étiquetés augmente de 0 à un nombre fixe lorsque tous les pixels recherchés sont trouves.

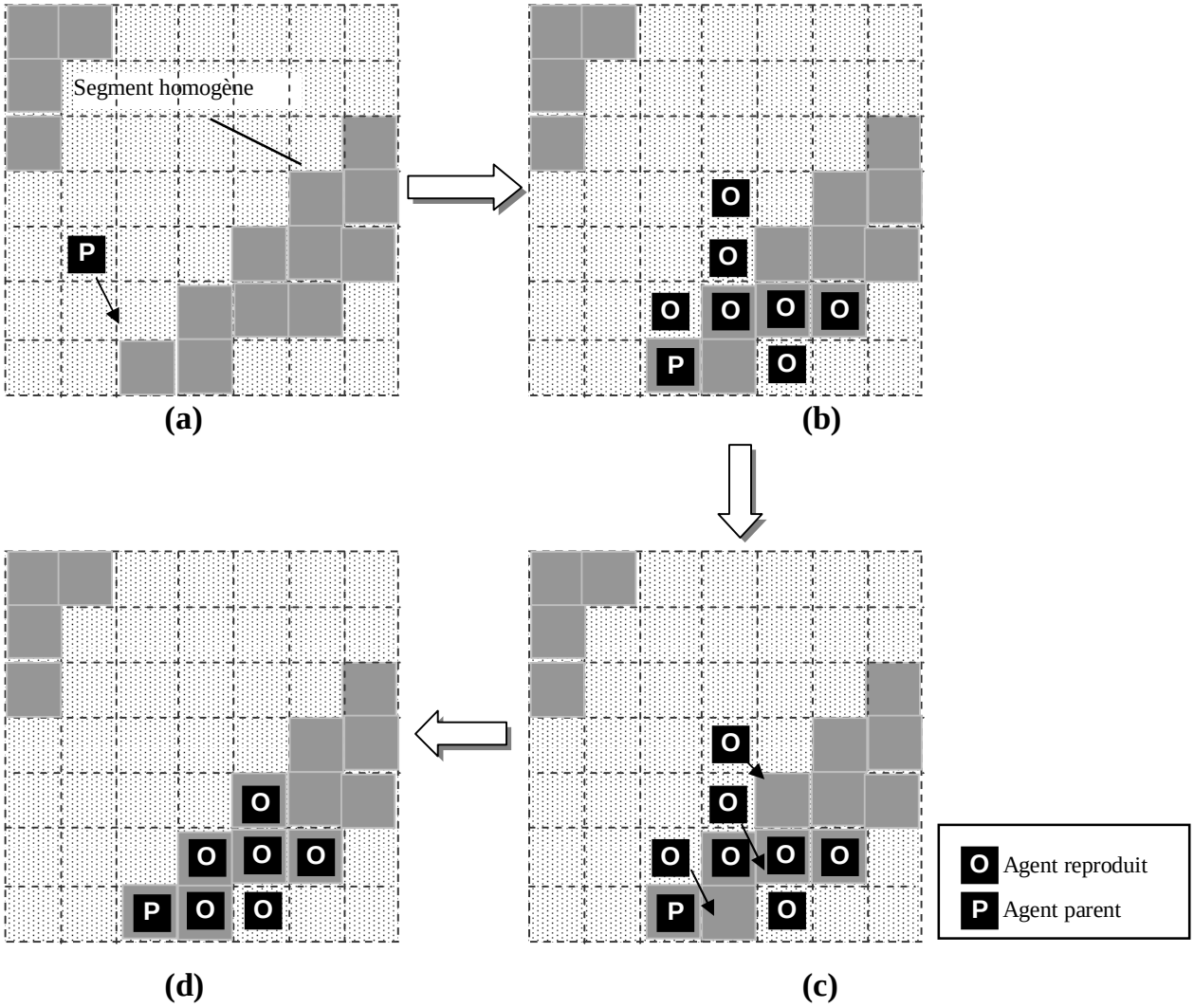


Figure 4.43 : ***Comportement de diffusion***

Disparition (Decay)

Si après un certain nombre d'itérations (durée de vie), un agent n'arrive pas à localiser une région homogène il sera retiré carrément de l'environnement. Ce comportement est donné par la formule:

DECAY : Si étapes > durée de vie.

Alors $a_{(i,j)}^{(t)} \Rightarrow null$

Le but d'introduction de ce comportement est d'éviter d'avoir un processus essai-erreur infini et de minimiser les calcules à effectuer.

Définition : à chaque agent α_i une fonction fitness est associée, noté $f(\alpha_i)$ tel que :

$$f(\alpha_i) = \begin{cases} 1 - \dfrac{\text{l'age de l'agent avant reproduction}}{\text{durée de vie de l'agent}} & \text{.... si } \alpha_i \text{ trouve une région homogène} \\ -1 & \text{..... sinon} \end{cases}$$

La fonction fitness prend une valeur maximale 1, lorsque l'agent est placé directement sur un pixel appartenant à la région homogène.

4.3.6.2.4 Les interactions

Dans notre système, il n'existe pas d'interactions directes entre entités, mais des interactions indirectes en partageant l'information permettant le choix de direction de diffusion. Une entité interagit avec son voisinage (niveau de gris) et agit sur l'environnement en étiquetant un pixel.

4.3.6.2.5 Mise à jour du vecteur de comportement

Dans notre contribution à l'amélioration d'algorithme de segmentation proposé par Jiming Liv, notre population d'agent est organisée sous forme de famille d'agents. Chaque famille étant constituée d'un agent parent et ses descendants. Le but de cette organisation est de répartir la tâche d'apprentissage. C'est-à-dire que la mise à jour du vecteur de comportement d'agent dépend seulement de celui des vecteurs des agents de sa famille. Le calcul des éléments du vecteur de comportement de l'agent (α_i) est donné par la formule :

$$Q_{(q \in \Theta)} = \frac{\sum_{j=1}^{N} Q_{(q,j)}}{N}$$

avec : Θ : L'ensemble des directions possibles.

N : Nombre d'agents de la même famille de (α_i) ayant une fonction fitness ≥ 0.

$Q_{(\theta,j)}$: probabilité de la direction de diffusion θ pour l'agent (α_i).

La mise à jour du vecteur de comportement permet à un agent de sélectionner les directions les plus efficaces ce qui optimise le processus de recherche de segments homogènes.

4.3.6.3 Phase 3 (Mesure des performances du système)

Un processus de segmentation est dit performant, lorsque tous les pixels appartenant à la région homogène recherchée sont localisés et étiquetés.

Pour évaluer les performances de notre système on a utilisé un ensemble d'images de référence avec leurs images segmentées correspondantes (Etalon de comparaison).
Une fonction d'évaluation (fitness) est introduite, représentant la qualité de segmentation réalisée, ceci en comparant l'image segmentée avec l'image préalablement segmentée.
La fonction fitness est donnée par la formule :

$$F = \frac{\text{Nombre de pixels correctement étiquetés}}{\text{Nombre de pixels étiquetés d'image étalon}}$$

La fonction fitness F prend la valeur optimale 1 si tous les pixels concernés sont étiquetés, et tend vers 0, lorsque le nombre de pixels correctement étiquetés est très petit.

4.3.7 Stratégie de recherche

Initialement les agents sont répartis aléatoirement sur la représentation 2-D d'une image à niveaux de gris. L'age de chaque agent est initialisé à 1. Pour donner plus de liberté aux agents, on va les permettre de choisir aléatoirement la direction de diffusion tant que leur age n'a pas dépassé un certain seuil. Si ce seuil est dépassé, et l'agent n'arrive pas à localiser un segment homogène, son vecteur de comportement est mis à jour pour lui permettre de trouver rapidement et efficacement un segment homogène. En se basant sur les vecteurs de comportement des membres de sa famille ayant une fitness ≥ 0.

Un agent devient parent s'il vient de localiser une région homogène, donc il va reproduire un certain nombre d'agents identiques ayant le même comportement initial ci-dessus. L'agent parent devient inactif, et sa vie sera prolongée pour permettre aux agents reproduits de bénéficier de son expérience. Si un agent n'arrive pas à localiser un segment homogène et que son age a atteint la durée de vie prédéfinie, il sera définitivement éliminé du système.

Un comportement de migration est nécessaire dans notre système car l'accumulation des agents reproduits dans une même région va ralentir l'exécution, donc lors de processus de reproduction, les agents doivent être diffusés vers des pixels qui ne sont pas occupés par d'autres agents et qui ne sont pas étiquetés.

4.3.8 Algorithme de segmentation

Entrée : A : Image numérique à niveau de gris.
Sortie : Régions homogènes étiquetées.

```
Début :
    Lecture d'image
    Initialisation des paramètres
    Création d'une population d'agent {α}^(0) répartis aléatoirement sur l'image.
    A ← {α}^(0)
    Tant que  A ≠ϕ  faire
        Pour  Chaque α ∈ A  faire
            mettre à jour le vecteur de comportement
            activer le stimulus local.
            Si l'agent α rencontre un pixel appartenant à une région homogène.
            Alors
                reproduire un ensemble d'agents {α_i}^(g+1)
                A ← A ∪ {α_i}^(g+1)
                désactiver α : A ← A - α
                étiqueter le pixel trouvé
                migration
            Sinon
                diffuser
            finsi

            Si  (age_α > durée de vie) et (état ≠ parent) alors
                désactiver α : A ← A - α
                éliminer l'agent du système
            Sinon
                Age_α ← Age_α + 1
            Fin
        Fin pour
    Fin tantque

    Afficher les régions homogènes.
Fin
```

4.3.9 Implémentation

4.3.9.1 Choix de langage

Pour l'implémentation de notre système complexe nous avons opté vers la plate forme Net logo. Ce logiciel possède le premier avantage d'être conçu en JAVA (langage orienté objet), ce qui le rend portable sur n'importe quelle machine. D'autre part, il utilise une extension du langage LOGO ce qui permet à un utilisateur non informaticien de développer des programme sans connaissance des langages informatiques classiques. En outre il est relativement simple à manipuler et possède des outils intégrés permettant un traitement aisé des résultats comme les illustrations graphiques. Mais l'avantage principal de ce langage c'est qu'il permet la

modélisation d'un système complexe composé d'un nombre infini d'entités autonomes situées dans un environnement statique ou dynamique, et fonctionnant en parallèle .

Ainsi il permet la simulation des interactions entre entités (par exemples les interactions entre fourmis dans une colonie de fourmis). Finalement on peut dire que NetLogo permet la modélisation des comportements des entités autonomes au niveau micro et le comportement émergent au niveau macro.

4.3.9.2 Présentation de Net logo

Net logo se compose d'une seule fenêtre depuis la quelle sont accessibles différentes interfaces via des onglets. Net logo permet de manipuler trois types d'entités : les tortues "Turtles" qui sont les éléments principaux, les patchs qui sont des éléments discrétisés de l'environnement et un observateur (observer) considéré comme contrôleur central du monde virtuel. Il permet de gérer les programmes et de passer des commandes spécifiques aux tortues et aux patches. Toutes les procédures sont écrites via l'interface « Procédures » et seront exécutées soit grâce à des boutons que l'on peut crées par le biais de l'interface graphique, soit en tapant leur nom dans le centre de commande

Chacune des entités autonomes représentées par tortue possède des commande qui lui sont propre.

L'environnement dans lequel évoluent les tortues est un monde continu en forme d'espace de deux dimension/3D divisé en patchs

4.3.9.3 Exemple d'illustration : segmentation d'image médicale

Pour valider notre approche, nous avons choisi une image médicale IRM cérébral (Figure 4.44), il s'agit d'une image complexe à 256 niveau de gris et de taille 512 x 512. Quatre classes d'agents sont utilisées, chaque classe est responsable à localiser un type prédéfini de segment homogène (voir table 4.6). Initialement, pour chaque classe un certain nombre d'agents est distribués aléatoirement sur l'image.

Figure 4.44 : ***Image à segmenter***

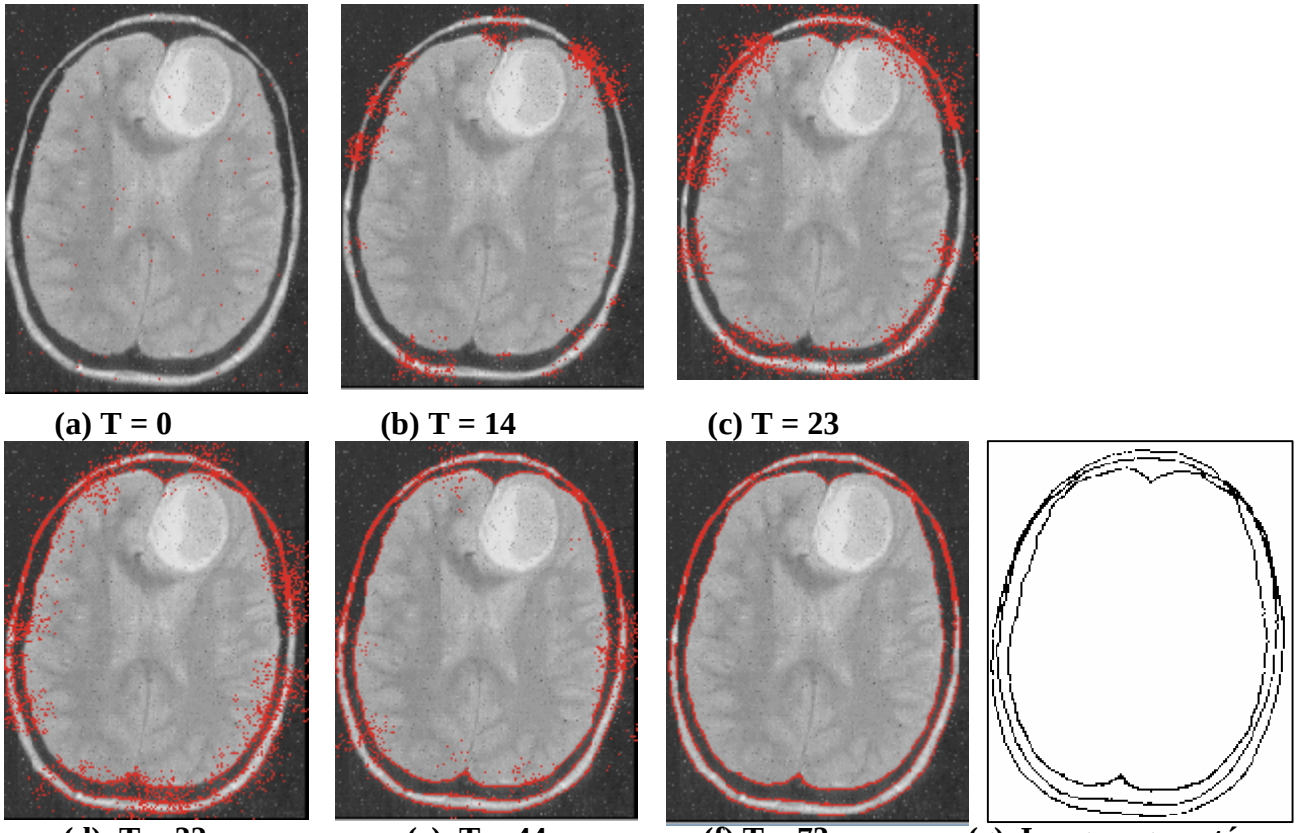
Attribut	Valeurs			
Classe	**Classe 1**	**Classe 2**	**Classe 3**	**Classe 4**
Type de segment	Contour	Région	Région	Région
δ	-	-	-	10
η_1	-	-	-	3
η_2	-	-	-	8
M_1	90	200	155	130
M_2	122	230	180	155
D_1	-	-	-	0
D_2	-	-	-	3
Durée de vie	10	10	8	10
Nombre de progénitures	3	3	2	2
Rayon de diffusion et reproduction	1	1	1	1
$R_{(i,j)\text{-région}}$	1	1	1	1

Table 4.6 : ***Classes de segmentation et leurs attributs***

4.3.9.4 Discussion

Initialement les classes d'agents sont utilisées séparément, c'est-à-dire à un moment donnée, une seule classe d'agents constituée de 100 agents est distribuée aléatoirement sur la représentation graphique 2D de l'image à segmenter. Les figures 4.45, 4.46 et 4.47 montre les différentes étapes de segmentation des classes 1, 2 et 3 et le résultat final est obtenu respectivement après 73, 29 et 69 étapes

Classe 1 :

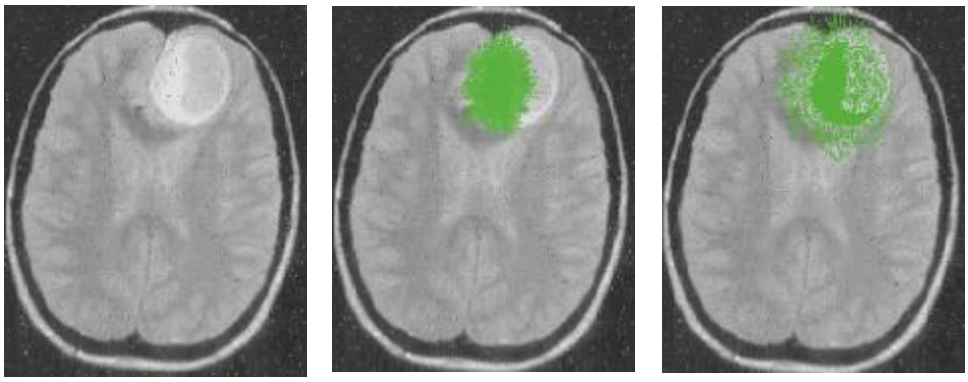


(a) T = 0 **(b) T = 14** **(c) T = 23**

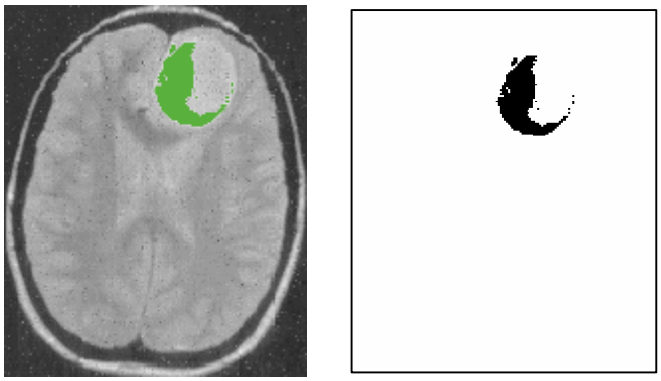
(d) T = 32 **(e) T = 44** **(f) T = 73** **(g) Image segmentée**

Figure 4.45 ***: étapes de segmentation (classe 1)***

Classe 2 :



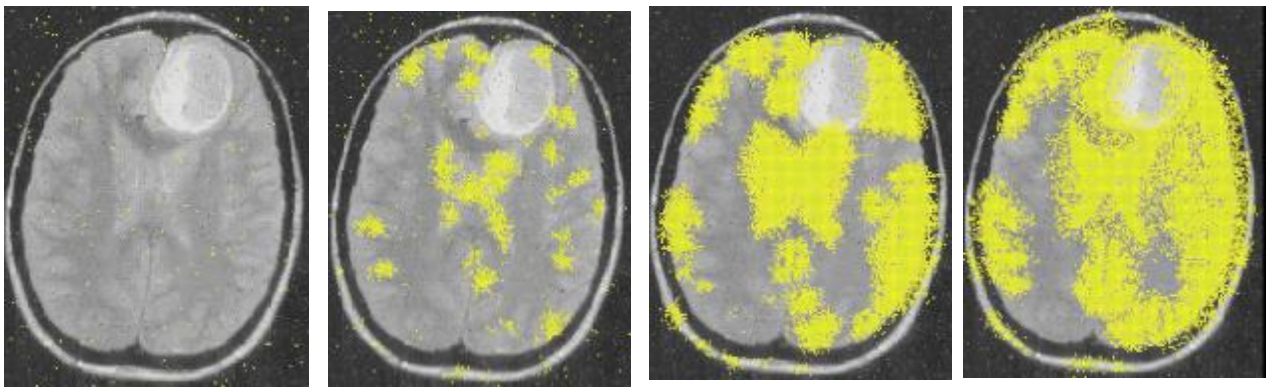
(a) T = 0 (b) T = 6 (c) T = 18



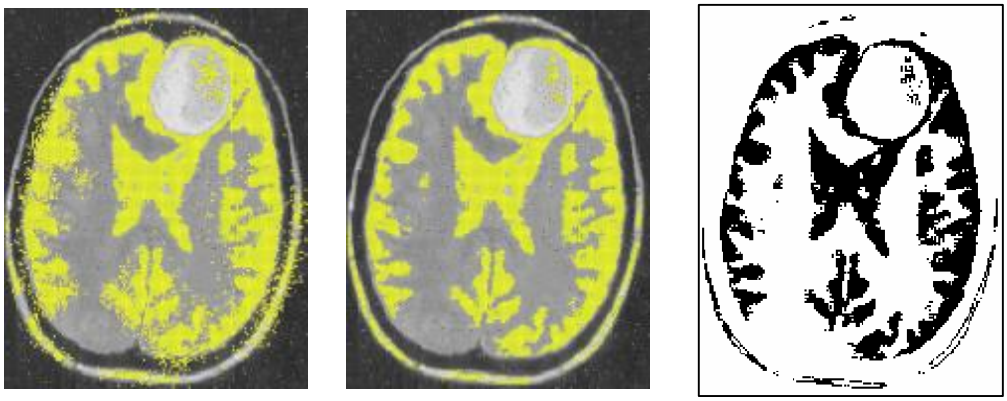
(d) T = 29 **(e) Image segmentée**

Figure 4.46 ***: étapes de segmentation (classe 2)***

Classe 3 :



(a) T = 0 **(b) T = 6** **(c) T = 9** **(d) T = 15**



(e) T = 21 **(f) T = 69** **(g) Image segmentée**

Figure 4.47 : étapes de segmentation (classe 3)

Les figures 4.48, 4.49 et 4.52 représentent le nombre d'agents actifs en fonction de périodes de temps, pour les classes 1,2 et 3.

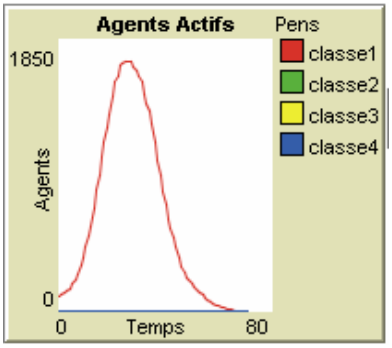


Figure 4.48 ***représentation des agents actifs de la classe 1 en fonction de périodes de temps***

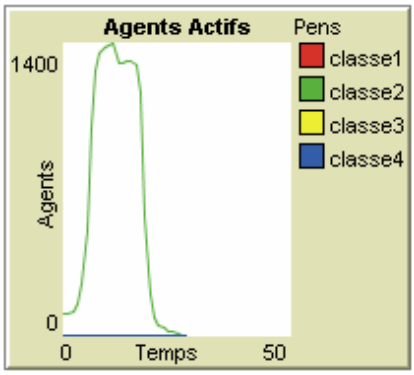


Figure 4.49 **: Représentation des agents actifs de la classe 2 en fonction de périodes de temps**

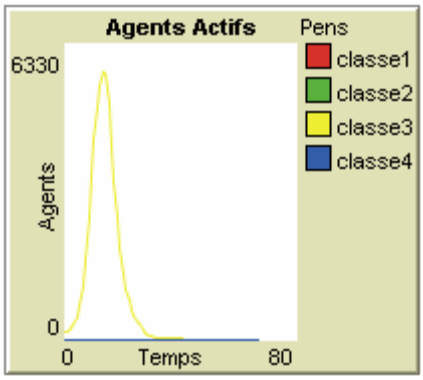
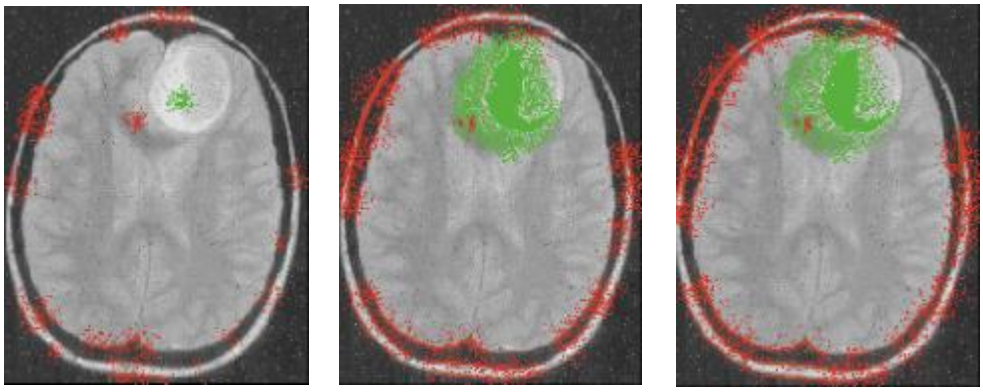
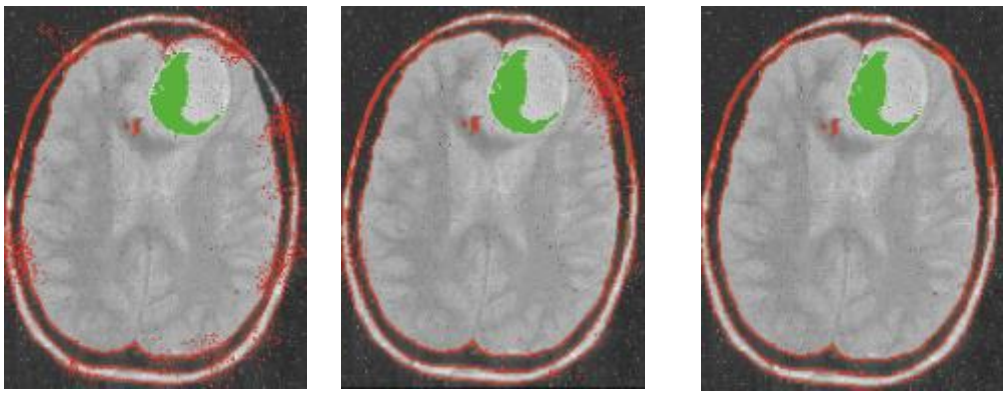


Figure 4.50 **: Représentation des agents actifs de la classe 3 en fonction de périodes de temps**

En deuxième lieu, nous avons exploité le parallélisme de notre système et par conséquent, nous avons introduit les différentes classes d'agents sur la représentation graphique de l'image pour rechercher simultanément les régions homogènes prédéfinies.(Figure 4.51)



(a) T = 14 **(b) T = 23** **(c) T= 27**



T = 39 **T = 50** **T = 65**

Figure 4.51 ***: Exemple d'utilisation de deux classes d'agents simultanément***

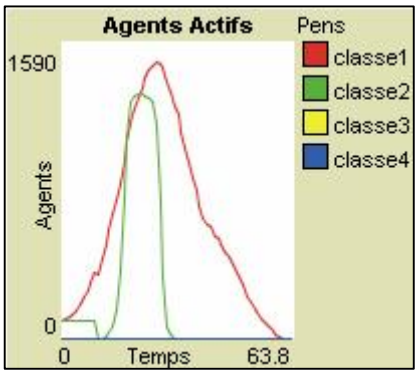


Figure 4.52 ***Représentation des agents actifs de la classe 1 et 2 en fonction de périodes de temps***

Les améliorations qu'on a apporté à cette approche proposé par jiming liu nous ont permet d'avoir des résultats satisfaisants à savoir :

- Un nombre d'agent réduit est utilisé pour effectuer cette tâche.
- Le parallélisme du système est bien exploité
- Le processus d'apprentissage des agents est réparti
- La recherche des segments homogènes est optimale.

Notre contribution se résume en :

- L'organisation des agents en famille d'agents : cela permet des mises à jour locales des vecteurs de comportements (diffusion)
- L'introduction du principe de déplacement aléatoire : pour donner plus de liberté aux agents afin de s'adapter avec l'environnement dynamique.
- L'introduction du processus de migration : pour éviter une concentration des agents actifs.

4.4 Deuxième étape

4.4.1 Principe

La deuxième partie de notre travail est consacrée à l'optimisation des paramètres généraux du processus de segmentation d'image déjà présenté à savoir : le nombre des entités autonomes (Agents) utilisés, leur durée de vie, nombre de progénitures et le rayon de diffusion.

La complexité de la tâche, nous ramène à utiliser les algorithmes génétiques pour optimiser ces paramètres.

Un ensemble d'images de référence avec leurs images segmentées correspondantes sont utilisés pour l'entraînement.

Une fonction d'évaluation (fitness) qui présente le critère d'évaluation de la qualité de segmentation réalisée en donnant à chaque fois des valeurs pour les différent paramètres de segmentation, ceci en comparant l'image segmentée avec l'image préalablement segmentée (Etalon de comparaison).

La fonction fitness est donnée par la formule :

$$F = \frac{\text{Nombre de pixels étiquetés correctement}}{\text{Nombre de pixels étiquetés d'image étalon}}$$

Cette fonction prend une valeur maximale 1, si tous les pixels concernés sont étiquetés.

Chaque solution du problème est représentée par un chromosome (Figure 4.53).

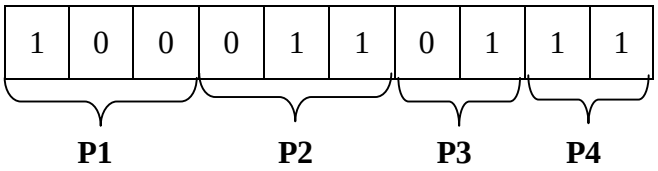


Figure 4.53 ***: Représentation d'un chromosome***

Avec :

P1 : Paramètre représentant la taille de la population d'agents répartis initialement sur la représentation 2D de l'image à segmenter (l'environnement).

P2 : Durée de vie d'un agent.

P3 : Nombre de progénitures lors d'une opération de reproduction.

P4 : Rayon de diffusion (pas).

Les valeurs possibles des différents paramètres sont représentées la table :

Code	Valeur
0 0 0	50 Agents
0 0 1	100 Agents
0 1 0	150 Agents
0 1 1	200 Agents
1 0 0	250 Agents
1 0 1	300 Agents
1 1 0	350 Agents
1 1 1	400 Agents

Code	Valeur
0 0 0	1
0 0 1	2
0 1 0	3
0 1 1	4
1 0 0	5
1 0 1	6
1 1 0	7
1 1 1	8

Code	Valeur
0 0	1
0 1	2
1 0	3
1 1	4

Code	Valeur
0 0	1
0 1	2
1 0	3
1 1	4

Table 4.7 ***: Valeurs possibles des paramètres à optimiser***

La taille du chromosome est de 10. Donc on a $2^{10} = 1024$ solutions possibles.

4.4.2 Gestion de l'évolution

On démarre avec une population initiale de taille fixe. L'évolution de cette population est gérée un algorithme génétique. Le principe général est de sélectionner les meilleurs individus en fonction de la valeur de la fonction d'évaluation (fitness) qui permet à la fin de chaque génération, d'évaluer l'adéquation des paramètres proposés. Un groupe est ensuite formé avec les meilleurs individus afin qu'ils se reproduisent. Des mutations sont possibles avec une certaine probabilité lors de la reproduction. Puis l'évaluation est relancée avec la nouvelle population composée des "géniteurs" et des "progénitures".

Nous avons défini 4 procédures pour notre algorithme génétique :

- La première sert à exécuter la tâche principale de segmentation d'images pour une population donnée.
- La deuxième procédure sert à sélectionner les géniteurs en calculant la valeur de la fonction d'évaluation (fitness). En fixant notre seul à 1, nous ne retenons ainsi que des individus issus de la meilleure moitié de la population autrement dit on veut que l'ensemble des reproducteurs soit égal à la moitié de la population du départ. Dans ce cas à la fin de chaque génération, le programme trie notre population selon la valeur de fitness de chaque individu et on retient que les individus ayant une valeur de fitness au dessus de la moyenne.
- La troisième procédure sert à créer de nouveaux individus à partir du pool des reproducteurs. Sur les principes de la génétique, les génomes des "parents" vont subir des recombinaisons pour donner les nouveaux génomes des "enfants". Chaque parent ne peut se reproduire qu'une seule fois et chaque opération de reproduction donne naissance à deux enfants. Puisque le pool des géniteurs est égal à la moitié de la population, le nombre total d'individus (parents + enfants) va rester stable.
- Enfin, la quatrième procédure de notre algorithme génétique consiste à réaliser des mutations lors de la reproduction. Chaque enfant possède une chance sur 100 d'être muté. La mutation consiste à remplacer un bit quelconque du génome.

4.4.3 Discussion

Les résultats suivants sont obtenus en exécutant notre algorithme génétique sur une population d'individus représentant chacune les paramètres à optimiser pour réaliser la tâche principale qui consiste à déterminer les régions homogènes dans une image à niveaux de gris. Nous nous intéressons aux agents de la classe 1.

§ D'après la table [4.8] les meilleurs résultats sont réalisés à partir d'une population initiale de 150 agents, cela signifie que le nombre nécessaire d'agent pour réaliser cette tâche doit être de préférence au dessous de la moyenne, ce qui représente une optimisation des ressources mémoires (Figure 4.54).

§ Le nombre optimal de progéniture est soit 4, soit 3 , cela se traduit par le fait de se retrouver dans la région homogène, il est préférable que le nombre de progéniture soit au dessus de la moyenne, ce qui permet d'avoir une population active des agents dans une région qui a plus de chance d'avoir les caractéristiques de la région homogène recherchée (Figure 4.55).

§ Le nombre total des agents utilisés au long du processus de segmentation varie entre 4800 et 6800 agents, représentant un nombre optimal par rapport à la taille réelle de l'image qui de 512 x 512 pixels, qui contient alors 262144 pixels.

§ Pour le rayon de diffusion, il est préférable qu'il prend la valeur 1 ou 2, ce qui permet aux agents de bien explorer l'image et par conséquent visiter tous les pixels de l'image à segmenter.

§ Enfin, les meilleurs résultats sont obtenus, avec des agents ayant une durée de vie au dessus de la moyenne, ce qui permet aux agents de bien explorer l'image, ainsi permet la coopération entre les agents de la même famille par le biais de l'information représentée par le vecteur de comportement. L'objectif est d'avoir une durée d'apprentissage acceptable. Cela n'empêche pas d'avoir de bons résultats en diminuant la valeur de la durée de vie aux alentours de la moyenne et en augmentant les valeurs de nombres de progénitures et celle de rayon de diffusion (Figure 4.56)

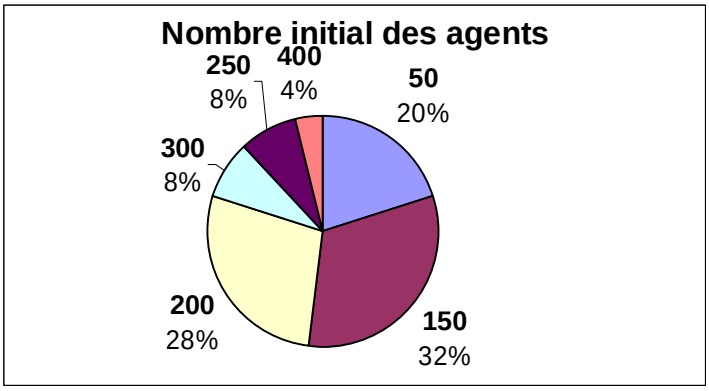


Figure 4.54 ***: Nombre initial des agents***

N°	Fitness	Nombre initial des agents	Durée De vie	Nombre de progéniture	Rayon de diffusion	Nombre total des agents	Durée de segmentation
1	**0.9982**	150	8	4	1	6850	216
2	**0.9964**	150	8	4	1	6338	273
3	**0.9869**	300	8	3	1	5268	104
4	**0.9851**	150	5	4	2	6762	126
5	**0.9827**	150	5	4	2	6746	134
6	**0.9815**	200	8	4	3	6788	228
7	**0.9809**	200	8	4	1	6784	203
8	**0.9797**	150	8	4	3	6726	240
9	**0.9779**	150	7	4	2	6714	188
10	**0.9774**	300	8	3	3	5223	114
11	**0.9750**	200	5	4	1	6744	120
12	**0.9738**	250	4	4	1	6786	81
13	**0.9726**	50	6	4	2	6578	234
14	**0.9696**	50	6	4	2	6558	169
15	**0.9607**	200	8	3	3	6698	117
16	**0.9583**	200	8	3	4	5024	115
17	**0.9571**	250	4	4	2	6674	88
18	**0.9487**	200	4	4	2	6568	89
19	**0.9482**	400	6	3	3	5173	78
20	**0.9476**	150	8	3	4	4920	118
21	**0.9476**	150	8	3	4	4920	118
22	**0.9333**	50	6	3	3	4748	90
23	**0.9267**	50	6	3	4	4715	92
24	**0.9184**	50	3	4	3	6214	68
25	**0.9178**	200	5	3	3	4820	66

Table 4.8 : ***Résultat du processus d'optimisation des paramètres***

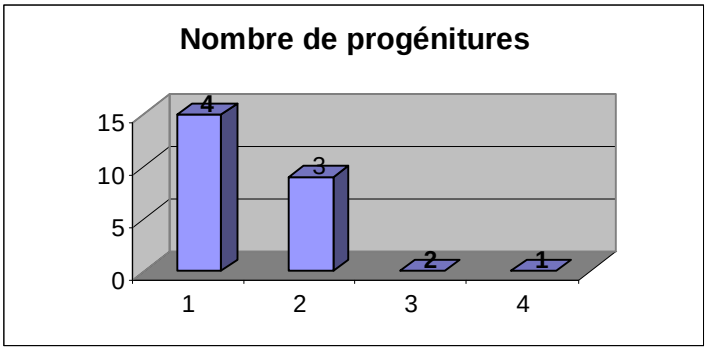


Figure 4.55 ***: Nombre de progénitures***

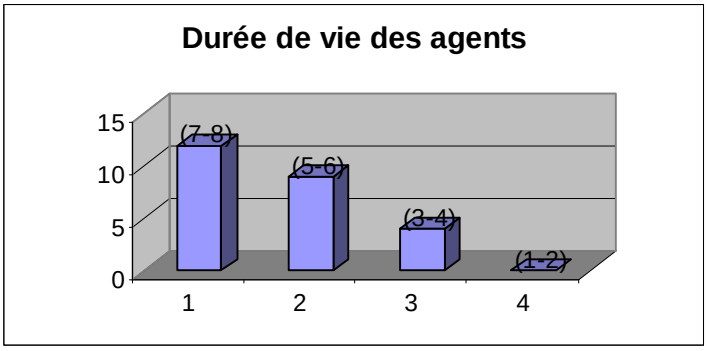


Figure 4.56 ***: Durée de vie des agents***

4.5 Troisième étape

Cette étape est basée sur l'approche A.O.C par auto découverte. Cette étape consiste à ajuster automatiquement les règles de comportement en fonction d'évolution de l'environnement. Dans ce cas on considère que les règles de comportement doivent êtres modifiées en fonction des changements subis dans l'environnement

L'idée c'est de permettre aux agents de s'adapter avec l'environnement en ajustant les paramètres des règles de comportement à savoir la durée de vie, le nombre de progénitures et ceci par le biais d'une coopération entre les agents de la même famille en partageant quelques informations.

Dans ce cadre on peut :

- Ajuster la durée de vie d'un agent en fonction de la valeur de sa fonction d'évaluation : si l'agent arrive à localiser un pixel appartenant à la région homogène recherchée avec une certaine valeur de fitness, sa durée de vie sera augmentée pour permettre à ces agents descendants d'exploiter son expérience.
- Ajuster le nombre de progénitures lors d'une reproduction selon l'environnement : par diminution de ce nombre s'il existe une forte concentration des agents actifs au même emplacement, soit par amplification s'il s'agit d'une faible concentration des agents actifs.
- Ajuster le pas de diffusion, en augmentant sa valeur si l'agent n'arrive pas après certaines itérations de localiser un pixel appartenant à la région homogène recherchée.

CONCLUSION GENERALE

Conclusion générale

Actuellement, il est de plus en plus difficile - sinon impossible - de contrôler correctement l'activité de ces logiciels situés dans des environnements de plus en plus dynamiques. Pour faire face à ces difficultés, une solution consiste à laisser plus d'autonomie aux logiciels afin qu'ils s'adaptent au mieux aux imprévus et donc concevoir des systèmes multi-agents adaptatifs. En effet, les systèmes multi-agents classiques prennent bien en compte la complexité des interactions, mais la dynamique n'est que très peu présente.

Il s'agit de développer des systèmes capables de travailler "en intelligence" avec leur environnement : les agents interagissent dans un environnement incertain et dynamique, le système doit être capable de s'auto-observer afin de détecter les phénomènes émergents et adapter son comportement et sa structure à ces phénomènes et à l'évolution de son environnement. Ou d'une manière plus précise, développer des systèmes informatiques adaptatifs.

Dans ce contexte, nous avons introduit une approche ascendante appelée A.O.C Autonomy Oriented Computing comme nouveau paradigme permettant de caractériser les comportements émergents des systèmes complexes ainsi la résolution des problèmes difficiles. Ceci en considérant l'autonomie comme le noyau de modélisation des comportements dans n'importe quel système complexe, et en s'inspirant des phénomène d'autonomie et d'auto-oganisation dans la nature.

Nous avons choisi le domaine de traitement d'images et plus précisément la segmentation d'image à niveaux de gris, comme application de cette méthodologie de résolution de problèmes difficiles. Les résultats sont satisfaisants, du fait qu'on peut généraliser notre approche sur les images couleurs.

Nos travaux futurs porteront sur l'utilisation de cet nouveau paradigme dans d'autres domaines de recherches à savoir : les réseaux mobiles et sans fil et l'optimisation des ressources distribuées sur le Web.

BIBLIOGRAPHIES

[1] S. Hassas, Systèmes complexes à base de Multi-agents situés, LIRIS-Novembre **2003**

[2] K. Dooley , Complex Adaptive Systems : A nominal definition

[3] Stuart Russell et Peter Norvig : Artificial Intelligence : a modern approach. Prentice Hall, 1ère édition, Janvier **1995**

[4] Michell Wooldridge et Nicols Robert Jennings : Intelligent agent. Artificial intelligence , **1995**

[5] Jacques Ferber : Les systèmes Multi-agents : vers une intelligence collective. Inter-editions, **1995**

[6] J.P Müller, Méthodologie de conception de systèmes multi-agents, de résolution de problèmes par émergence. Hermès, Novembre **1998**

[7] Z. Guessoum, Modèles et architecture d'agents et de systèmes Multi-agents adaptatifs. Laboratoire d'informatique de Paris 6, Décembre **2001**

[8] C. Boutilier. Planing, learning and coordination in multiagent decision processes. In Proceedings of the 6th conference on theoretical Aspects of Rationality and Knowledge, The Netherlands **1996**

[9] C. Watkins et P. Dayan . Q-learning. Machine learning ,**1992**

[10] Olivier Buffet, Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs, Université Henri pointcaré Nancy **2003**

[11] T.D Barfoot, G.M.T D'Eleuterio, An evolutionary approach to Multiagent Heap Formation, University of Toronto Canada **1999**

[12] Jiming Liu, Shiwu Zhang, and Jie Yang, Charactrizing Web Usage Regularities With Information Foraging Agents , Avril **2004**

[13] Jiming liu, Yuan Y. Tang , Adaptive Image Segmentation with Distributed Behavior-Based Agents, Hong Kong Baptist University, Juin **1999**

[14]Jiming Liu, K. Ching Tsui, Jianbing Wu, Introducing Autonomy Oriented Computation,**2001**

[15] Jiming Liu, Xiaolong Jin, Kwok Ching Tsui, Autonomy Oriented Computing (AOC) : Formulating Computational Systems with Autonomous Components

[16] Kwok Ching Tsui, Jiming Liu, Evolutionary Multi-agent Diffusion Approach to Optimisation, Hong Kong Baptist University, décembre **2001**

[17] Jiming liu, Y.Y Tang, Y.C Cao, An evolutionary autonomous agents approach to image feature extraction,juillet **1997**

[18] W.S. Mc Cullogh and W. Pitts, "A logical calculus of the ideas immanent in nervous activity ", Bulletin of Mathematical Biophysics 5, p.115-133, **1943**.

[19] D.O. Hebb, "The Organisation of Behaviour", Wiley, New York, **1949**

[20] F. Rosenblatt, "The Perceptron : a Probabilistic Model for Information Storage and Organisation in the Brain", Psychological Review, p. 386-408, **1958**.

[21] P.C. Mitiguy and T.R. Kane, "Motion Variables Leading to Efficient Equations of Motion", The International Journal of Robotics Research, Vol. 15, No. 15, pp. 522-532, October **1996**.

[22] D.E. Rumelhart and J.L. Mc Clelland, "Parallel Distributed Processing", The MIT Press, vol. 1 et 2, Cambridge, **1986**.

[23] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", Proceedings of the National Academy of Sciences, p. 460-464, **1982**.

[24] M. Schoenauer : Algorithmes évolutionnaires : exemples d'application, en optimisation combinatoire

[25] T. M. Mitchell, "Machine learning", Livre, Mc Graw-Hill International Editions, **199**7.

[26] S.Wolfram : Statistical mechanics of cellular automata, Reviews of modern physics, 55:601-644, **1983**

[27] S.Wolfram : Universality and complexity in cellular automata, Reviews of modern physics, **1984**

[28] S.Wolfram : Problems in the theory of cellular automata, Physica script, T9:170-183, Reviews of modern physics, **1985**

[29] Imed Jarras, Brahim Chaib-draa : Aperçu sur les systèmes Multi-agents, Montréal, Juillet **2002**

[30] Philipes Blangi : Etat de l'art sur les plates-formes et les langages multi-agents appliqués aux écosystèmes, Master de recherche, Laboratoire d'analyse des systèmes du littoral, Année universitaire 2004-2005, Université du littoral (France)

[31] Peter Habermehl, Cours Intelligence Artificielle **2004-2005**, Site Web : http://www.liafa.jussieu.fr/~haberm/cours/ia/

[32] ALEXIS DROGOUL, De la simulation multi-agent à la résolution collective de problèmes - Une étude de l'émergence de structures d'organisation dans les systèmes multi-agents, Thèse de Doctorat de l'Université Paris VI présentée le 23 Novembre **1993**.

[33] Michel Occello : Méthodologie et architectures pour la conception des systèmes multi-agents, Université Joseph Fourier Grenoble, Juillet **2003**

[34] T. Xavier, R. Christine, G. Marie-Pierre, G. Pierre, Comportements individuels adaptatifs dans un environnement dynamique pour l'exploitation collective de ressource, I.R.I.T Toulouse **1999**.

[35] Benjamin DRIEU, L'intelligence artificielle distribuée appliquée aux jeux d'équipe situés dans un milieu dynamique : l'exemple de la RoboCup , Université Paris 8, 11 Octobre **2001**

[36] L. Bonte, Représentation multi-échelle pour plates-formes à grands nombres d'agents, IFL Juin **2005**

[37] Agnès Guillot, Systèmes adaptatifs naturels et artificiels : l'approche Animat, Site Web : http://animatlab.lip6.fr

[38] Ph. Lucidarme, Apprentissage et adaptation pour des ensembles de robots réactifs coopérants, Université Montpellier II, Novembre **2003**

[39] T D Barfoot and G M T D'Eleuterio, Learning Distributed Control for an Object-Clustering Task, University of Toronto, Institute for Aerospace Studies Technical Report, Mars **2003**.

[40] Jacques Ferber. Vers une intelligence collective. InterEditions, **1995**.

[41] Jiming Liu, Autonomy Oriented Computing (AOC): From problem solving to complex system modeling, Hong Kong Mai **2005**