IM

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie *Houari Boumédiene*

Faculté d'Electronique et d'Informatique
Département d'Informatique

Mémoire de Fin d'Etudes
Pour l'obtention du diplôme
De Magister en Informatique

Option : Intelligence Artificielle et Bases de Données Avancées

Une extension multi-agents du problème de La poursuite de Cliff & Miller

L'approche *COCAGE*

Collective Co-evolution based on Multi-Agents Environnement Approach

Etudié par :

– TAIBI Amine

Soutenu le : 13 Février 2005
Devant le jury composé de :

– Mr S. LARABI	Maître de Conférence USTHB	Président
– Mme H. DRIAS	Professeur USTHB / INI	Directeur de thèse
– Mr Y. DUTHEN	Professeur Toulouse 1	Co-directeur de thèse
– Mr A. DJERRADI	Maître de Conférence USTHB	Examinateur
– Mr. H. AZZOUNE	Maître de Conférence USTHB	Examinateur

2004-2005

Remerciements

Je souhaite en premier lieu exprimer ma profonde gratitude à Mme Drias Habiba qui ma accordé sa confiance pour plusieurs fois, et surtout sa patience envers ses étudiants et leurs problèmes.

Je veux ensuite adresser mes plus vifs remerciements à Mr Yves Duthens, qui m’a donné l’occasion de travailler dans un domaine très vivant.

Je remercie Mr Slimane LARABI, qui me fait l'honneur de présider ce jury.

Je tiens aussi à exprimer ma reconnaissance envers Monsieur A. Djeradi et Monsieur H. AZZOUNE qui ont accepté d'être des examinateurs dans ce jury.

Plusieurs personnes ont bien voulu consacrer un peu de leur temps à corriger les fautes de français du manuscrit de ce mémoire, je les remercie, ainsi tous mes amis, et les membres de ma famille, qui ont contribués de prés ou de loin pour que ce mémoire puisse voir le jour.

Résumé

Dans le cadre de ce travail on a réalisé une étude bibliographique sur les algorithmes génétiques et les stratégies évolutionnistes ainsi que le codage génétique, les réseaux de neurones et les contrôleurs neuronaux et la création génétique des contrôleurs sensor-moteur, le problème de la poursuite de Cliff & Miller un classique de la co-évolution; une notion bien définie dans notre étude. Après une étude de quelques propositions et extensions du problème de la poursuite, on a essayé de proposer une architecture générique **COCAGE** (*Collective Co-evolution based on Multi-Agents Environnement Approaches*) pour le problème de la poursuite comme cas d'analyse et pour tout les problème concernant la co-évolution dans les milieux compétitifs. Comme conclusion, on constate que la co-évolution collective est présente dans tous les modèles de la vie artificielle et la vie organique, ce qui donne une bonne ébauche pour qu'elle soit utilisée dans tout processus de développement informatique et autre.

Sommaire

Introduction Générale

Le problème de la *poursuite* de Cliff & Miller est devenu un classique de la théorie de la *co-évolution* artificielle, ce problème consiste en un modèle simple de *simulation* de scénario de poursuite, entre deux espèces, un *poursuiveur* et un *poursuivi*, appelé parfois *prédateur* et *proie*. La simulation se résume en trois phases, la première est la création des deux *espèces* poursuiveurs et poursuivis, dans cette phase on génère la morphologie et le comportement - structure et contrôle – des individus, la deuxième phase consiste à choisir un individu de chaque espèce et les mettre dans un monde de simulation pendant une période de temps fixe, et finalement, on évalue chaque individu pour le scénario de la poursuite réalisée. Après ces trois phases, on sélectionne les meilleurs individus de chaque espèce et on reprend la simulation infiniment, c'est ce processus qui est la *co-évolution* connue dans la biologie.

Le travail qui a été demandé, consiste à voir en détail le problème de la poursuite de Cliff & Miller comme première étape, la deuxième étape est de voir les extensions possible du problème mais dans le même contexte simple-agent, comme dernière étape proposer des extensions dans un contexte multi-agents ce qui a menu à proposé une approche multi-agents qui est **COCAGE**, i.e. « *Collective Co-evolution based on Multi-Agents Environnement Approach* », cette dernière généralise le problème de la poursuite de Cliff & Miller dans deux dimensions, la première est l'agentification des deux espèces et la deuxième est la communication entre agents de même espèce. Avant de passer à la structure du mémoire, il faut insister sur quelque points, le problème de la poursuite est un problème très pratique on, le trouve dans des simulations de systèmes biologiques, des missiles anti-avions ou anti-missiles, la réalité virtuelle, la vie artificielle, les systèmes GPS des satellites et enfin dans des applications plus conviviales les jeux vidéo.

Le travail est jalonné sur sept chapitres, qui respecte les phases de la poursuite de Cliff & Miller et les extensions proposées.

Le premier chapitre, introduit les approches génétiques et plus exactement, les algorithmes génétiques, i.e. AG, l'importance des AG pour le problème de la poursuite est le fait que les individus sont créés à base d'un algorithme génétique approprié.

Les chapitres deux, trois et quatre introduisent les approches neuronales qui sont les réseaux de neurones récurrents « *RNR* », réseaux de neurones récurrents dynamique « *RNRD* », et les contrôleurs neuronaux. Ces concepts sont au cœur du contrôle de comportement des entités de la poursuite de Cliff & Miller. On introduit aussi les techniques d'évolution génétique des contrôleurs neuronaux, ainsi que leurs codage et enfin quelques algorithmes d'apprentissage des réseaux de neurones.

Le chapitre cinq introduit le problème de la poursuite de Cliff & Miller en détails, commençant par un passage sur la co-évolution, l'évolution et la course aux armements. On passe à l'explication des méthodes de simulation qui consiste en deux étapes, la modélisation physique des entités et le modèle neuronal du contrôle. Pour modéliser l'évolution des entités Cliff & Miller on utilise un algorithme génétique spécial avec un codage spécial, ce dernier est aussi introduit. Comme deuxième partie, on est intéressé à donner quelques extensions du modèle qui répondent à quelques critiques et limitations trouvées. Ces extensions sont soit structurelles ou organisationnelles.

Le chapitre six donne un état de l'art sur les Systèmes Multi-Agents, i.e. *SMA*, la raison d'introduire les SMA est l'extension multi-agents envisagée pour le problème de la poursuite de Cliff & Miller, dans ce chapitre on trouve la définition d'un agent, d'un système multi-agents, de la société d'agents, de la communication entre agents et de plusieurs autres notions. En résumé ce chapitre peut être lu séparément grâce à une autonomie dans ces concepts.

Enfin, le chapitre sept représente une proposition d'une approche Multi-Agents pour l'extension du problème de la poursuite de Cliff & Miller appelée COCAGE i.e. « *Collective Co-evolution based on Multi-Agents Environnement Approach* ». Le chapitre commence par une agentification des entités de la poursuite, qui est un modèle d'agent hybride avec trois niveaux qui sont : l'Agent Cognitif, l'Agent réactif et l'Agent Maker, pour la phase d'agentification. En deuxième phase, on passe à la définition du modèle de système multi-agents qui concrétise la poursuite de Cliff & Miller étendue. Cette dernière phase, étend la poursuite à deux dimensions, l'homogénéité et la communication entre agents.

Une conclusion générale est donnée enfin avec des perspectives et des extensions de l'approche.

Les approches Génétiques

1. Introduction

L'évolution biologique a engendré des systèmes vivants extrêmement complexes. Elle est le fruit d'une altération progressive et continue des êtres vivants au cours des générations et s'opère en deux étapes : la sélection et la reproduction.

- La sélection naturelle est le mécanisme central qui opère au niveau des populations, en sélectionnant les individus les mieux adaptés à leur environnement.

- La reproduction implique une mémoire : l'hérédité, sous la forme de gènes. Ce matériel héréditaire subit, au niveau moléculaire, des modifications constantes par mutations et recombinaisons, aboutissant ainsi à une grande diversité.

- Ces principes, présentés pour la première fois par Darwin, ont inspiré bien plus tard les chercheurs en informatique. Ils ont donné naissance à une classe d'algorithmes regroupés sous le nom générique d'Algorithmes Evolutionnaires (ou Evolutionary Algorithms (EA)). Les sections suivantes en présentent les fondements.
-

2. Algorithmes évolutionnaires

Il existe une catégorie de problèmes pour lesquels il est difficile, voire impossible, de trouver une solution en un temps limité. Il est alors utile de trouver une technique permettant la localisation rapide de solutions sous optimales, sachant que l'espace de recherche a une taille et une complexité suffisamment importantes pour éliminer toute garantie d'optimalité. Pour cela, un système capable de s'auto-modifier au cours du

temps, tout en améliorant sa performance dans l'accomplissement des tâches auxquelles il est confronté, semble ouvrir la voie à une recherche intéressante.

Les EAs sont basés sur des principes simples. En effet, peu de connaissances sur la manière de résoudre ces problèmes sont nécessaires, même si certaines peuvent être exploitées afin de rendre plus efficace l'évolution (en effet, il n'est pas réaliste d'espérer obtenir une méthode d'optimisation raisonnablement efficace si aucune connaissance sur le domaine à traiter). C'est pourquoi, dans de nombreux domaines, les chercheurs ont été amenés à s'y intéresser.

Un certain nombre de travaux ont porté sur l'optimisation de fonctions mathématiques complexes telles celles proposées dans [De Jong, 1975]. Les EA ont été appliqués à différents problèmes issus de la recherche opérationnelle : la coloration de graphes [Eiben et al.,1998], le problème du voyageur de commerce [Tao et Michalewicz,1998] [Watson et al.,1998], la gestion d'emploi du temps [Paechter et al., 1998], etc.

On trouve aussi des applications en robotique pour la recherche de trajectoire optimale [Ahuactzin et al.,1995] ou l'évitement d'obstacles [Schultz et Grefenstette , 1992], en vision pour la détection [Heap et Samaria, 1995] [Baluja, 1998] ou la reconnaissance d'images [Liu et Wechsler, 1998] [Brouard et al. ,1998], en recalage d'images médicales [Roux et Jacq,1993], en reconnaissance de formes [Iwata et al.,1990], mais aussi en mécanique pour l'optimisation de formes [Kane et Schoenauer, 1997], en chimie [Geyer et al., 1998], économie et gestion de production [Fontanili et Vincent ,1997], gestion de trafic aérien [Oussedik et Delahaye, 1998], etc.

Par ailleurs, pour se rapprocher des phénomènes trouvés dans la nature, divers travaux ont porté sur des individus "intelligents" basés sur des réseaux connexionnistes. Ils ont montré l'intérêt d'appliquer les EA à des réseaux de neurones [Belew et al., 1990] [Whitley, 1995], et particulièrement dans l'association avec l'apprentissage par rétro-propagation. Ils permettent entre autres de choisir un bon ensemble de poids initiaux, ce qui permet non seulement d'améliorer la vitesse de convergence des réseaux, mais aussi d'éviter de les faire converger vers des états correspondant à des minima locaux qui ne donneraient pas des solutions optimales.

Dans la suite de ce chapitre, nous présentons les principes généraux des systèmes adaptatifs basés sur les EA, ainsi que les méthodes qui en découlent.

Principes généraux

Les EA sont une classe d'algorithmes d'optimisation par recherche probabiliste basés sur le modèle de l'évolution naturelle. Ils modélisent une *population* d'*individus* par des points dans un espace. Un individu est codé dans un *génotype* composé *de gènes*

correspondant aux valeurs des paramètres du problème à traiter. Le génotype de l'individu correspond à une solution potentielle au problème posé, le but des EA est d'en trouver la solution optimale.

La plupart des problèmes peuvent être résolus par une méthode de type recherche locale. C'est le cas, par exemple, de l'apprentissage d'un réseau de neurones par rétro-propagation du gradient. Partant d'une configuration de poids initiale, la méthode va chercher la meilleure solution dans le voisinage de cette configuration.

Cette solution est optimale localement mais peut ne pas correspondre à un optimal global car il est possible qu'il existe une meilleure solution qui n'est pas dans le voisinage de la configuration initiale. Le Schéma 1 illustre ce type de problème.

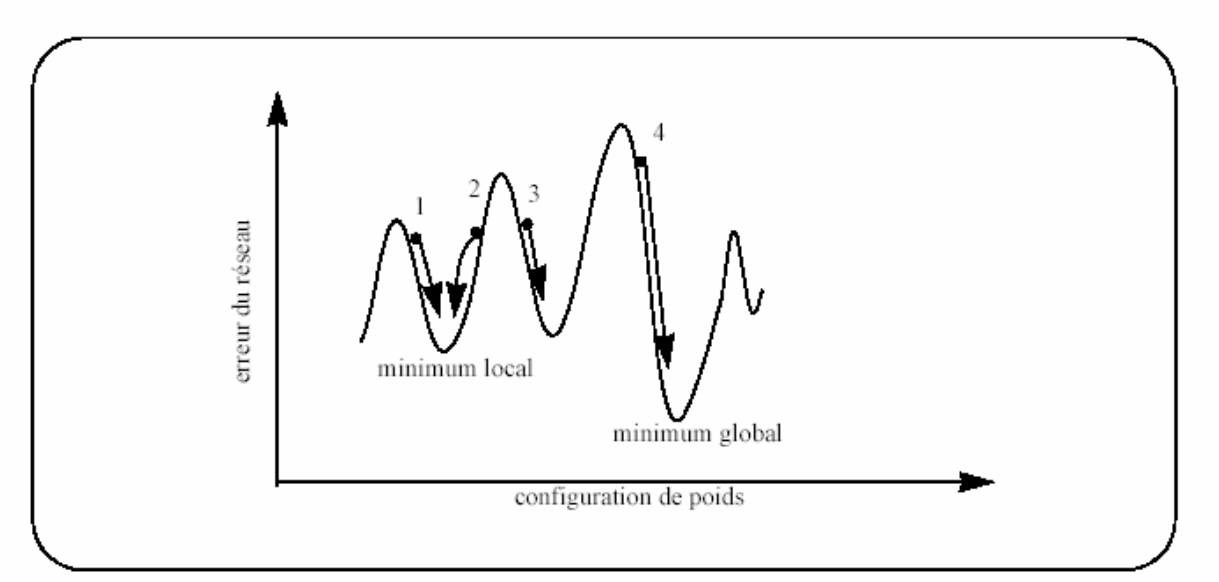


Schéma 1 : Exemple de convergence locale ou globale d'un réseau de neurones.
Convergence vers des erreurs correspondant à des minima locaux (configurations 1, 2 et 3) ou à un minimum global (configuration 4).

Les EA ont montré leur capacité à éviter la convergence des solutions vers des optima locaux, aussi bien lorsqu'ils sont combinés avec des méthodes de recherche locale comme la rétro-propagation du gradient [Belew et al.,1990] que lorsqu'ils sont seuls [Goldberg, 1989].

Quel que soit le type de problème à résoudre, les EA opèrent selon les principes suivants : la population est initialisée de façon dépendante du problème à résoudre (l'*environnement)*, puis évolue de génération en génération à l'aide d'opérateurs de *sélection*, de *recombinaison* et de *mutation*. L'environnement a pour charge d'évaluer les individus en leur attribuant une performance (ou *fitness*). Cette valeur favorisera la

sélection des meilleurs individus, en vue, après reproduction (opérée par la mutation et/ou recombinaison), d'améliorer les performances globales de la population.

Plusieurs types d'évolution ont été développés, donnant naissance à trois grandes tendances : les Algorithmes Génétiques (ou *Genetic Algorithms* (GA)), les Stratégies d'Evolution (ou *Evolution Strategies* (ES)) et la Programmation Evolutive (ou *Evolutionary Programing* (EP)). Une branche annexe, la Programmation Génétique (ou *Genetic Programing* (GP)) peut aussi rentrer dans ce type de systèmes.

De ces quatre méthodes classiques ont dérivé différentes techniques mélangeant les méthodes d'évolution des unes et des autres. Impossible à classer dans l'une des quatre familles citées ci-dessus, elles sont néanmoins considérées comme des EA.

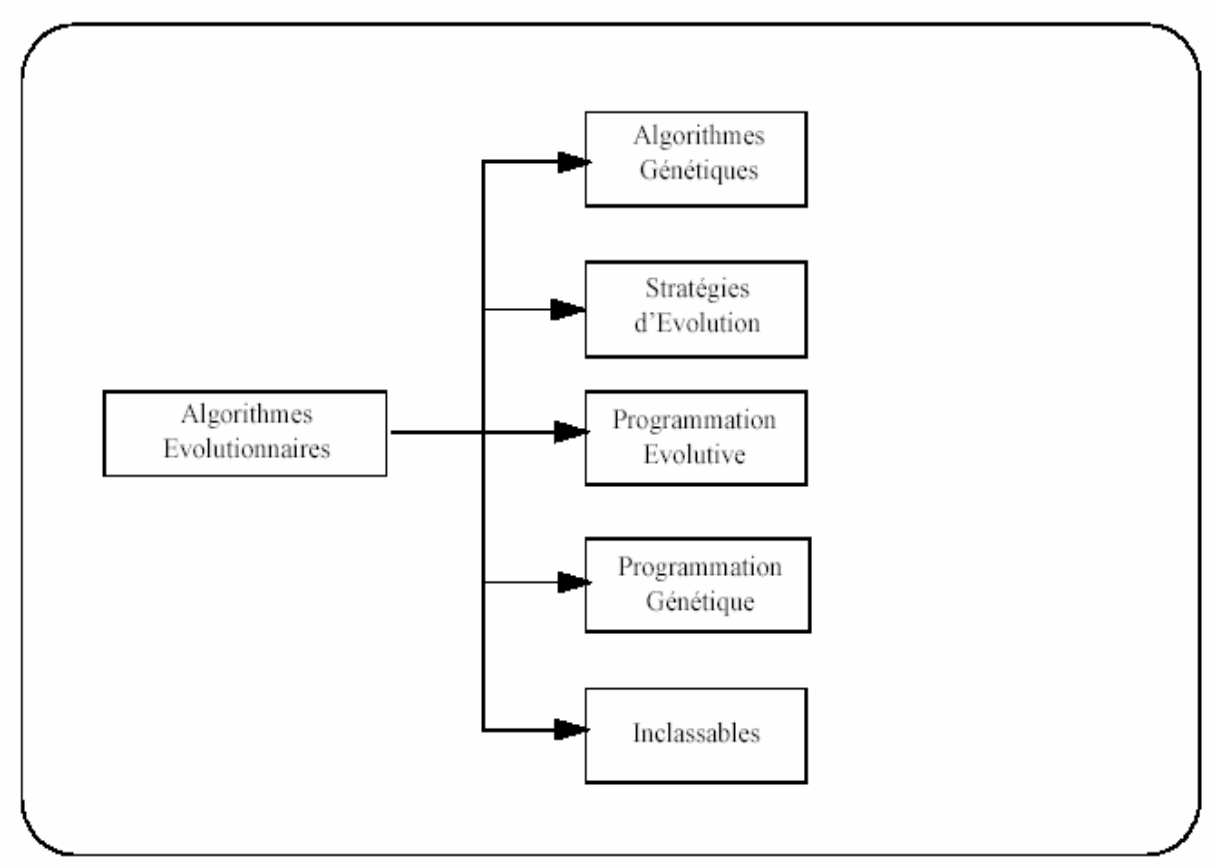


Schéma2. Différentes branches des algorithmes évolutionnaires.

3. Les Algorithmes Génétiques

3.1 Introduction

Par définition même l'étude de l'économie se base sur l'optimisation, et les deux mots sont souvent associés. La sophistication des modèles économiques conduit de plus en plus souvent à des problèmes d'optimisation toujours plus complexes. Le seul contexte des modèles dynamiques nous fournit bon nombre de problèmes de résolution ou d'estimation. Les études économiques s'en trouvent limités, et le modélisateur doit alors incorporer des contraintes techniques pour faire face à ces problèmes. Cette situation est quelque peu paradoxale, si l'on considère l'augmentation des moyens et des puissances de calcul disponibles

Un grand nombre de méthodes économétriques, comme les moindres carrées généralisés et le maximum de vraisemblance, reposent sur la maximisation d'une fonction souvent complexe. Parmi toutes les méthodes d'optimisation, les méthodes de gradient sont les plus exigeantes en matière de conditions nécessaires de convergence. Ce sont des méthodes d'optimisation locales, qui connaissent une grande popularité. Pour le maximum de vraisemblance, Cramer , recense les désagréments possibles lors de l'application de cette méthode. L'algorithme de maximisation peut ainsi ne pas converger en un temps acceptable, il peut s'éloigner et donner des valeurs infinies pour certaines composantes des paramètres, il peut également boucler, et revenir sans cesse au même point, etc. Ces algorithmes d'optimisation conventionnels (de la famille du gradient ou de Newton-Raphson) sont des algorithmes grimpeurs qui se basent sur l'évaluation préalable du gradient, la pente, pour déterminer la direction de recherche de l'optimum. Goffe et al. [Goffe et al ,1994] comparent cette situation à celle d'un aveugle cherchant le sommet d'une montagne. La connaissance du terrain passe alors uniquement par ses pieds. Pour peu que le terrain soit régulier et le point de départ bon, il atteindra le sommet. Toutefois ces deux conditions sont rarement simultanément réalisées. Avec beaucoup de chance, c'est à dire avec une très bonne sensibilité et beaucoup de points de départs, il pourra atteindre le sommet pour autant que ce dernier soit unique. De plus, les hypothèses sur les fonctions à optimiser sont souvent très fortes, et ne sont pas vérifiées dans la pratique.

A l'inverse, les techniques de recherche aléatoires pures ne requièrent aucune hypothèse particulière sur la fonction d'évaluation et explorent l'espace sans tirer partie des propriétés de la fonction objective. Les algorithmes génétiques et le recuit simulé se situent entre ces deux extrêmes.

Une des particularités séduisantes des algorithmes génétiques et du recuit simulé, réside dans l'absence d'hypothèses particulière sur la régularité de la fonction objective. Aucune hypothèse sur la continuité de cette fonction n'est requise, ses dérivées

successives ne sont pas nécessaires, ce qui rend très vaste le domaine d'application de ces algorithmes. En outre ces algorithmes sont aisément parallélisables ce qui renforce leur efficacité dans le cadre de problèmes d'optimisation sur des espaces de dimension importante.

Les premiers travaux sur les algorithmes génétiques ont commencé dans les années cinquante lorsque plusieurs biologistes américains ont simulé des structures biologiques sur ordinateur. Puis entre 1960 et 1970, John Holland [Holland ,1975] , sur la base des travaux précédents, développa les principes fondamentaux des algorithmes génétiques dans le cadre de l'optimisation mathématique. Malheureusement, les ordinateurs de l'époque n'étaient pas assez puissants pour envisager l'utilisation des algorithmes génétiques sur des problèmes réels de grande taille. L'ouvrage de Goldberg [Goldberg, 1989] qui décrit l'utilisation des algorithmes génétiques dans le cadre de résolution de problèmes concrets a permis de mieux faire connaître ces derniers et a marqué le début d'un nouvel intérêt pour ces techniques.

Le peu d'hypothèses requises sur la fonction objectif permet de traiter des problèmes très complexes. La fonction objective peut ainsi être le résultat d'une simulation. On peut même imaginer, pour régler certains paramètres de l'algorithme génétique lui-même tels que la taille de la population, les différents pourcentages de croisement et de mutation, d'utiliser un algorithme génétique. La rapidité de convergence du premier devenant ainsi fonction d'évaluation du second.

Ces méthodes ne se réduisent cependant pas à la simple recherche d'optima d'une fonction, et s'avèrent être également de puissants outils pour l'analyse de situations dynamiques complexes comme l'on en rencontre sur les marchés financiers, ou en théorie des jeux. On peut ainsi modéliser les comportements d'agents par des suites d'éléments binaires correspondant à des stratégies et examiner la survie et l'évolution de ces agents, c'est à dire l'émergence de certaines stratégies.

Ces méthodes ne sont apparues que très récemment dans la littérature économique et économétrique. Pourtant les modèles économétriques actuels génèrent bon nombre de problèmes de maximisation sur des espaces de dimensions grandissantes. Dorsey et Mayer [Dorsey et Mayer ,*1995*]ont ainsi étudié récemment onze problèmes économétriques classiques en utilisant six procédures de maximisation différentes et leurs conclusions sont élogieuses envers les algorithmes génétiques. Andreoni et Miller [Andreoni et Miller, *1995*] ont utilisé cet outil pour organiser des enchères, remplaçant les agents par des algorithmes adaptatifs basés sur des algorithmes génétiques. De leur étude, quoique fort restrictive, résulte une meilleure connaissance des procédures menant à l'équilibre de Nash dans différents types d'enchères.

Holland et Miller [Holand et Miller, 1991] proposent d'étendre encore le champ de ces techniques par l'utilisation d'Agents Artificiels Adaptatifs (AAA). Ils concluent leur article ainsi :

"By executing these models (AAA) on a computer we gain (..) an opportunity to check the various unfolding behavior for plausibility, a kind of reality check. Whether or not agents behave in an optimal manner, the very act of contemplating such systems will lead to important questions and answers."

Nous proposons ici une présentation simple de ces nouveaux outils que sont les algorithmes génétiques et le recuit simulé. Notre but est d'expliquer les mécanismes généraux de ces algorithmes et d'en faire découvrir les usages et potentialités. La présente section est consacrée aux principes généraux qui animent ces algorithmes. Des illustrations mathématiques, économiques et économétriques sont présentés dans la troisième section . Nous proposons une synthèse des résultats théoriques les plus récents dans la quatrième section . Ces résultats sont apparus très tardivement, probablement à cause de la complexité induite par ces algorithmes et il faudra attendre 1993 pour qu'une démonstration complète et rigoureuse de convergence stochastique soit établie par R. Cerf [Cerf ,1994]. La cinquième section est consacrée aux perspectives qu'offrent ces outils dans les différents domaines de l'économie mathématique et de l'économétrie. En outre, nous donnons en appendice le traitement complet d'un exemple simple.

3.2 Principes

Les algorithmes génétiques sont des procédures qui s'inspirent des mécanismes de sélection naturelle et des phénomènes génétiques. Le principe de base consiste à simuler le processus d'évolution naturelle dans un environnement hostile. Ces algorithmes utilisent un vocabulaire similaire à celui de la génétique, cependant, les processus auxquels ils font référence sont beaucoup plus complexes.

On parlera ainsi d'individu dans une population. L'individu est composé d'un ou plusieurs chromosomes. Les chromosomes sont eux-mêmes constitués de gènes qui contiennent les caractères héréditaires de l'individu. Les principes de sélection, de croisement, de mutation introduits dans ce cadre artificiel, s'appuient sur les processus naturels du même nom.

Pour un problème d'optimisation donné, un individu représente un point de l'espace d'état. On lui associe la valeur du critère à optimiser. L'algorithme génère ensuite de façon itérative des populations d'individus sur lesquelles on applique des processus de sélection, de croisement et de mutation. La sélection a pour but de favoriser les meilleurs éléments de la population, tandis que le croisement et la mutation assurent une exploration efficace de l'espace d'état.

3.2.1 Principes généraux des algorithmes génétiques

Le mécanisme consiste à faire évoluer, à partir d'un tirage initial, un ensemble de points de l'espace vers le ou les optima d'un problème d'optimisation. L'ensemble du processus s'effectue à taille de population constante, que nous notons N. Par analogie avec la génétique, on parle alors de générations successives. L'ensemble du processus s'effectue à taille de population constante, que nous notons N; de sorte que les générations successives comportent toutes N individus.

Afin de faire évoluer ces populations de la génération k à la génération k +1, trois opérations (voir figure 1) sont effectuées pour tous les individus de la génération k :

- Une sélection d'individus de la génération k est effectuée en fonction du critère à optimiser ou plus généralement du critère d'adaptation au problème (fitness), on cherche ainsi à privilégier la reproduction des *bons* éléments au détriment des *mauvais*. Des opérateurs d'exploration de l'espace sont ensuite utilisés pour élargir la population et introduire de la nouveauté d'une génération sur l'autre.

- L'opérateur de croisement est appliqué avec une probabilité P_c à deux éléments de la génération k (parents) qui sont alors transformés en deux nouveaux éléments (les enfants) destinés à les remplacer dans la génération k + 1.

- Certaines composantes (les gènes) de ces individus peuvent ensuite être modifiés avec une probabilité P_m par l'opérateur de mutation. Cette procédure vise à introduire de la nouveauté dans la population

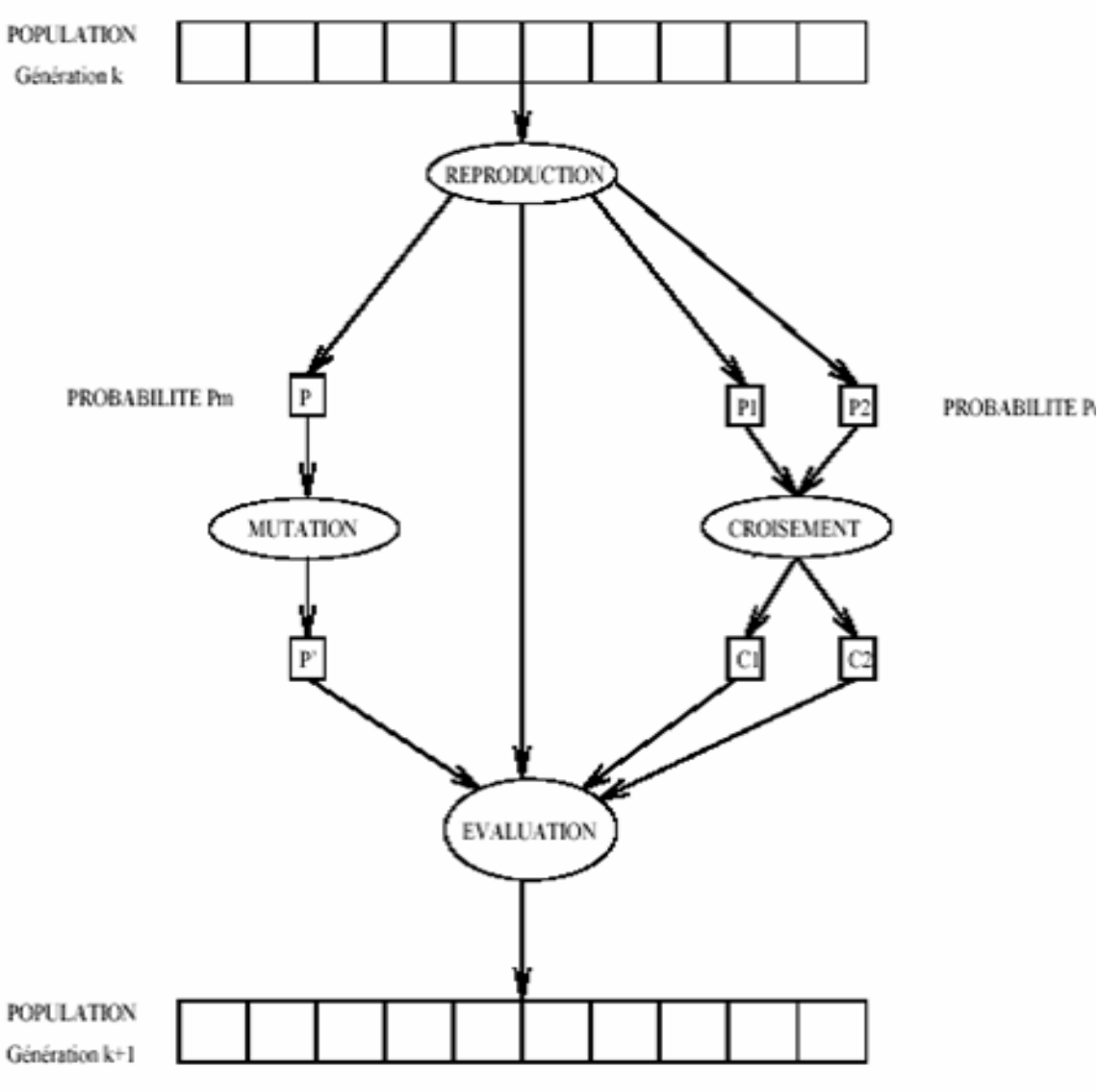


Fig. 1: Principe général des Algorithmes Génétiques

Cette procédure en trois points est ensuite renouvelée à taille de population constante. Les critères d'arrêts sont alors de deux natures :

- Arrêt après un nombre de générations fixé a priori. C'est la solution retenue lorsqu'un impératif de temps de calcul est imposé.
- Arrêt lorsque la population cesse d'évoluer ou n'évolue plus suffisamment rapidement, on est alors en présence d'une population homogène dont on peut penser qu'elle se situe à proximité du ou des optimums.

Il est à noter qu'a ce stade, aucune certitude concernant la bonne convergence de l'algorithme n'est assurée. Comme dans toute procédure d'optimisation l'arrêt est arbitraire, et la solution en temps fini ne constitue qu'une approximation de l'optimum.

Pour utiliser un algorithme génétique sur un problème particulier on doit disposer des cinq éléments suivants :

- Un principe de codage des éléments de l'espace admissible du problème, en éléments sur lesquels peuvent s'appliquer les trois opérateurs présentés ci-dessus. Ce codage intervient après une phase indispensable de modélisation mathématique du problème (voir figure 1). Le choix du codage des données dépend du problème traité et conditionne l'efficacité (vitesse de convergence, précision,..) de l'algorithme génétique.
- Un mécanisme de génération de la population initiale. Cette population initiale, qui sert de base aux générations futures, doit être la plus hétérogène possible.
- Une fonction d'utilité f, permettant de calculer l'adaptation de chaque élément au problème. Ce critère retourne une valeur de R+ appelée fitness.
- Des opérateurs permettant de diversifier et d'améliorer la population d'une génération sur l'autre ainsi que d'explorer le plus largement possible l'espace admissible.
- Des paramètres dimensionnels : taille de la population, critère d'arrêt, probabilités de croisement (P_c) et de mutation (P_m).

Il s'agit donc d'un *algorithme stochastique itératif* qui opère sur des ensembles de points codés, à partir d'une population initiale, et qui est bâti à l'aide de trois opérateurs : croisement, mutation, sélection. Les deux premiers sont des opérateurs d'exploration de l'espace, tandis que le dernier fait évoluer la population vers les optima du problème. Nous détaillons dans la section suivante les différentes phases de l'algorithme ainsi les principes de fonctionnement de ces opérateurs.

3.2.2. Codage et Opérateurs
Coder ou ne pas coder ?

Historiquement, les individus intervenant dans un algorithme génétique étaient codés sous forme de chaînes de bits5. Ce codage binaire contenant toute l'information nécessaire à la description d'un point dans l'espace d'état (voir également Alliot et Schiex [Alliot et Schiex, 1993]). Les opérateurs précités agissent alors sur les individus codés, c'est à dire sur de chaînes de bits. A titre d'exemple, considérons le problème de maximisation d'une fonction f(x) définie sur le domaine [0; 1], dont le maximum est atteint pour x*= 0:5.

Pour traiter ce problème, on associe les points du domaine [0; 1] à une chaîne de bits V dont la longueur P déterminera la précision de résolution. La chaîne V sera donc composée de P éléments binaires $V = (v_i)_{i=1,..,P}$ où $v_i \in \{0; 1\}$: Le point x_V correspondant sera défini par :

$$x_V = \sum_{i=1}^{P} v_i \cdot 2^{-(i-1)}$$

Ceci nous permet d'obtenir 2^P éléments différents dans l'intervalle [0; 1]; ce qui nous donne une précision de $1/(2^P - 1)$.

En examinant le codage au voisinage de 0,5, on constate que deux points très proches dans l'espace d'état peuvent être codés très différemment6.
En effet :

Variable	*Codage*
0.49999...	0111111111...
0.50000...	1000000000...

On évite cet inconvénient en utilisant un codage de Gray7.

Pour des problèmes d'optimisation dans des espaces de dimension supérieure, on concatène les chaînes de bits bout à bout. Par exemple, pour une fonction de deux variables *z = f(x; y)*, on code *x* et *y* sur leur domaine respectif puis on concatène *x* et *y* en une chaîne unique *xy*. Ce type de codage fonctionne bien mais présente l'inconvénient de perdre la structure du problème en fusionnant *x* et *y* dans une chaîne unique.

Il est également possible de ne pas coder les éléments de l'espace admissible du problème. Les opérateurs agissant directement sur les éléments de la population. Ainsi, les algorithmes génétiques utilisant des vecteurs réels étudiés par Golberg et Wright évitent ce problème en conservant les variables du problème dans le codage de l'élément de population sans passer par le codage binaire intermédiaire. L'exemple précédent serait codé à l'aide d'un vecteur à deux dimensions, on conserve ainsi la structure du problème dans le codage.

Nous verrons par la suite comment agissent les opérateurs sur des éléments codés sous forme de chaînes de bits, et sur des éléments réels.

3.2.3. Gestion des contraintes

La gestion des contraintes liées au problème est une tâche difficile et sensible pour laquelle l'utilisateur aura à arbitrer entre différentes techniques suivant son appréhension du problème. Ces contraintes sont de diverses natures et peuvent intervenir sur l'espace d'état (contraintes de signe, restrictions à un sous-espace, etc..), ou de manière plus complexe dans le problème lui même.

Dans le cas de contraintes sur l'espace dans lequel doit se faire la recherche, on pourra sélectionner les individus rapidement (sans avoir à les réévaluer) par différentes méthodes. Un individu « *hors champ»* pourra être :

- rejeté brutalement et remplacé par un autre individu tiré aléatoirement sur l'espace admissible ;
- ramené à la frontière la plus proche (principe du mur) ;
- reporté à la frontière diamétralement opposée à la frontière la plus proche (principe du tore).

Enfin il est bon de noter qu'il peut être préférable de garder des individus « *hors champ »* mais qui conservent une direction originale, plutôt que de confiner la population à un sous-espace.

Dans l'hypothèse où la gestion des contraintes ne peut se faire directement, les contraintes peuvent être incluses dans le critère à optimiser sous forme de pénalités. Ainsi, un individu qui viole une contrainte se verra attribuer une mauvaise fitness et sera donc éliminé, avec une forte probabilité, par le processus de sélection. Cette façon de gérer les contraintes est difficile. En effet, inclure les contraintes dans la fonction d'évaluation peut se faire de diverses façons, et un « *dosage* » s'impose pour ne pas favoriser la recherche de solutions admissibles au détriment de la recherche de l'optimum ou inversement. On risque alors de fournir une solution admissible certes, mais éloignée de l'optimum.

3.2.3. Génération aléatoire de la population initiale

Comme dans tout problème d'optimisation, une connaissance de *« bons »* points de départ conditionne la rapidité de la convergence vers l'optimum.

Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état, en veillant à ce que les individus produits respectent les contraintes.

Si par contre, des informations a priori sur le problème sont disponibles, il parait bien évidemment naturel de générer les individus dans un sous-domaine particulier afin d'accélérer la convergence.

Une nouvelle fois, les contraintes du problème pourront être incorporées (ou non) dans le tirage de la génération initiale.

Disposant d'une population d'individus non homogène, la diversité de la population doit être entretenue au cours des générations afin de parcourir le plus largement possible l'espace d'état, c'est le rôle des opérateurs de croisement et de mutation. Toutefois cette méthode diffère de la méthode de recherche aléatoire, puisque les générations successives doivent être évaluées et modifiées afin de converger, c'est ici qu'intervient l'opérateur de sélection.

3.2.4. Opérateur de sélection

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent. Les trois principes de sélection suivants ont retenu notre attention :

- Ordonnancement,
- Roue de la fortune,
- Roue modifiée,

Ordonnancement (Ranking)

C'est le principe de sélection le plus simple, il consiste à attribuer à chaque individu son classement par ordre d'adaptation. Le meilleur (c'est à dire ce- lui qui possède la meilleure fitness) sera numéro un, et ainsi de suite. On tire ensuite une nouvelle population dans cet ensemble d'individus ordonnés, en utilisant des probabilités indexées sur les rangs des individus. Cette procédure semble toutefois assez simpliste et exagère le rôle du meilleur élément au détriment d'autres éléments potentiellement exploitables. Le second, par exemple, aura une probabilité d'être sélectionné nettement plus faible que celle du premier, bien qu'il puisse se situer dans une région d'intérêt. Des procédures plus évoluées permettent de pondérer cette dominance des meilleurs éléments, c'est le cas des principes de roulette.

Le principe de la Roue de la fortune consiste à associer à chaque individu θ_i une probabilité P_i proportionnelle à sa fitness $f(\theta_i)$ dans la population.
Cette probabilité pourra être calculée comme :

$$P_i = \frac{S(f(\theta_i))}{\sum_{j=1}^{N} S(f(\theta_j))}$$

où S est une fonction régulière et croissante, au sens large.
Chaque individu est alors reproduit avec la probabilité Pi, certains individus (les « *bons* ») seront alors « *plus* » reproduits et d'autres (les « *mauvais* ») éliminés.

Remarque :

Dans la pratique, il est aisé de tirer N individus, affectés de la probabilité Pi; parmi N avec remise. Pour cela, on associe à chaque individu un segment dont la longueur est proportionnelle à sa fitness (ou à $S(f(\theta_i))$, plus exactement). On reproduit ici le principe de tirage aléatoire utilisé dans les roulettes de foire avec une structure linéaire. Ces segments sont ensuite concaténés sur un axe que l'on normalise entre 0 et 1 (voir figure 2). On tire alors un nombre aléatoire, de distribution uniforme entre 0 et 1, puis on « *regarde* » quel

est le segment sélectionné. Avec ce système, les grands segments, c'est-à-dire les bons individus, seront plus souvent adressés que les petits, on privilégie ainsi les individus ayant une forte fitness au détriment des individus moins forts, tout en gardant une structure aléatoire.

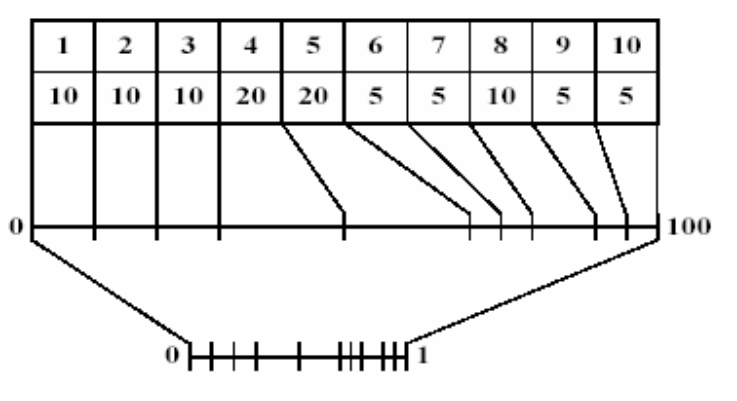


Fig. 2: Sélection de la roue de la fortune

Dans l'exemple présenté ci-dessous, la probabilité théorique de sélectionner l'individu θ_5 est de 20 pour cent.

Exemple :

Indices	1	2	3	4	5	6	7	8	9	10
Pop. initiale	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}
Fitness	3.2	0.5	0.2	1.5	2.5	0.3	0.2	0.4	1.5	0.3
Proba. P_i	0.30	0.04	0.01	0.14	0.23	0.02	0.01	0.03	0.14	0.02
Nouvelle pop.	θ_1	θ_4	θ_2	θ_6	θ_9	θ_5	θ_1	θ_9	θ_5	θ_7

Mais au regard de la faible dimension de cette population (10) on constate qu'il sera difficile d'obtenir cette espérance mathématique de sélection en raison du peu de tirages effectués. Un biais de sélection plus ou moins fort existe suivant la dimension de la population. Certains individus sont ainsi représentés plusieurs fois (θ_5, θ_1); tandis que d'autres disparaissent (θ_0, θ_3;::); d'autres enfin survivent « par chance », bien qu'ayant un adaptation faible (θ_7): La roue modifiée permet d'éviter ce genre de problèmes.

Décrivons ce principe de sélection à l'aide de l'exemple simple proposé ci-dessus (N = 10) :

Le principe consiste à créer un sous-tableau T_M un tableau de dimension M (M<N) tel que :

- Tous les individus ayant une fitness supérieure à la moyenne figurent dans T_M,
- Chaque individu θ_i est représenté N_i fois où N_i est la partie entière du rapport de la fitness à la moyenne des fitness μ,

$$N_i = Ent\left(\frac{f(\theta_i)}{\mu}\right)$$

Sur notre exemple, $\mu = 1,06$, ainsi l'individu aura 3 représentants dans ce tableau *(f(θ1)/ μ= 3,01)*. Pour notre exemple, T_M est de dimension $M = 7$:

Indices du tableau	1	2	3	4	5	6	7
Individus	θ_1	θ_1	θ_1	θ_4	θ_5	θ_5	θ_9

La création de ce tableau est purement déterministe. L'assurance d'un nombre précis de représentants pour la génération suivante élimine le biais du principe de sélection décrit précédemment. L'ajout d'un principe aléatoire permet de ne pas éliminer complètement les mauvais individus. En effet, dans la pratique, ces individus sont nécessaires car ils peuvent occuper des positions stratégiques pour l'obtention de l'optimum. Le tableau T_M est alors étendu à T_N de dimension N, de la façon suivante :

- Pour chaque individu θ_i; on calcule la partie fractionnaire du rapport de sa fitness à la moyenne (on obtient donc un nombre α_i entre 0 et 1).
- On tire ensuite aléatoirement μ entre 0 et 1; autant de fois qu'il manque d'individus, c'est à dire *(N - M)* fois.

  - Si $\eta < \alpha$, l'élément θ_i est ajouté dans le tableau T_M.
  - Sinon, on passe à l'indice i+1.

- Quand on arrive à l'élément d'indice *N* on repasse à l'élément 1.
  Sur notre exemple, T_M pourrait être complété de la façon suivante :

Indices du tableau	1	2	3	4	5	6	7	8	9	10
Individus	θ_1	θ_1	θ_1	θ_4	θ_5	θ_5	θ_9	θ_1	θ_3	θ_6

A cette étape du processus, on dispose déjà de _bons_ individus pour la nouvelle population, certains ont déjà été éliminés (comme θ_2, θ_7, et θ_{10}). Pour assurer la non homogénéité de la population, on effectue un brassage aléatoire du tableau d'indices T_N avant de reconstituer la nouvelle population et d'élargir la recherche à l'aide des opérateurs de croisement et de mutation.

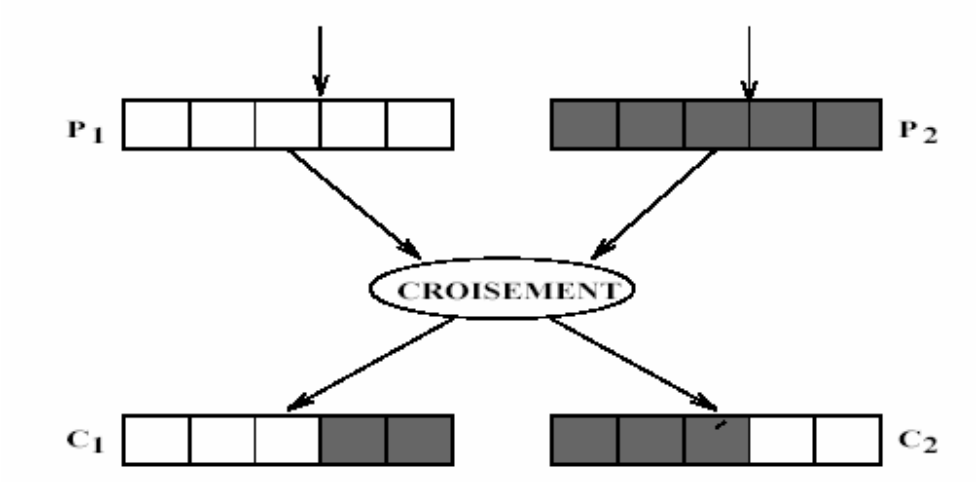


Fig. 3: Croisement chromosomique à un point

3.2.5. Opérateur de croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant les composantes des individus (chromosomes). Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants.

Initialement, le croisement associé au codage par chaînes de bits ou chromosomes, est le croisement à découpage de chromosomes (slicing crossover). Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position de découpage. On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes (les parents) P_1 et P_2, ce qui produit deux nouveaux chromosomes (les enfants) C_1 et C_2 (voir figure 4).

On peut étendre ce principe en découpant le chromosome non pas en deux sous chaînes mais en trois, quatre, etc.. Ce type de croisement est illustré par la figure 5.

Le croisement à découpage de chromosomes est très rapide à mettre en oeuvre lorsqu'on travaille sur des problèmes utilisant des codages entiers (binaires ou autres). Pour les problèmes où l'on utilise un codage réel, un croisement « barycentrique » est souvent utilisé. Deux parents P_1et P_2 sont sélectionnés ; deux nouveaux points sont créés sur la droite qui les relie créant ainsi C_1 et C_2 de la façon suivante (voir également la figure 5) :

$$\begin{cases} C_1 = \alpha P_1 + (1-\alpha) P_2 \\ C2 = (1-\alpha) P_1 + \alpha P_2 \end{cases}$$

où α est un coefficient de pondération aléatoire adapté au domaine d'extension des gènes.

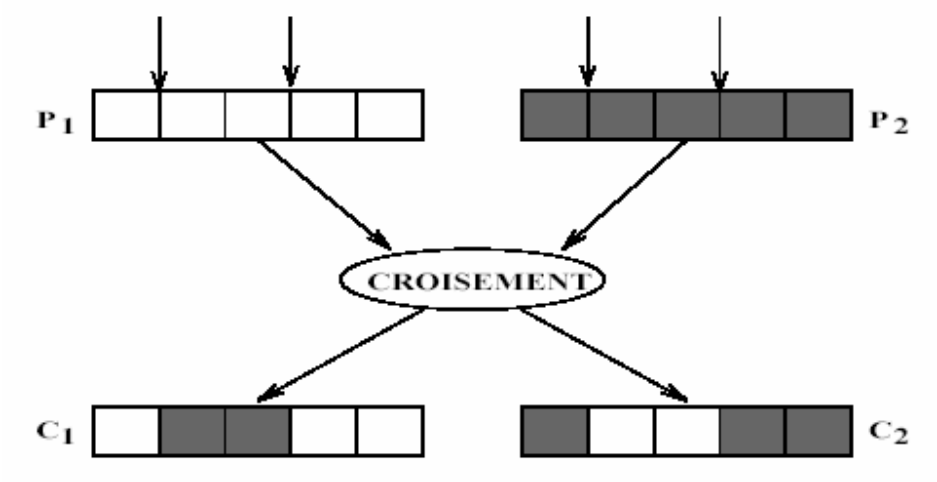


Fig. 4: Croisement chromosomique à deux points

Le croisement barycentrique permet de générer des points entre, ou à l'extérieur des deux éléments considérés. Toutefois ce type de croisement peut connaître des limitations importantes si les individus se situent sur la même droite (ou plus généralement dans le même sous espace). L'opérateur de croisement, utilisé seul, ne permet alors pas de rechercher les nouveaux éléments en dehors de cette droite. Dans le cas particulier d'un chromosome matriciel constitué par la concaténation de vecteurs, on peut étendre ce principe de croisement aux vecteurs constituant les gènes. Deux composantes $\overrightarrow{P_1}(i) et \overrightarrow{P_2}(i)$ sont sélectionnées dans chacun des parents à la même position i. Ils définissent deux nouveaux éléments par pondération. On crée ainsi $\vec{C}_1$(i) et $\vec{C}_2$(i) de la façon suivante :

$$\begin{cases} \overrightarrow{C_2}(i) = \alpha \overrightarrow{P_1} + (1-\alpha)\vec{P}_2(i) \\ \overrightarrow{C_2}(i) = (1-\alpha)\overrightarrow{P_1} + \alpha \vec{P}_2(i) \end{cases}$$

On peut imaginer et tester des opérateurs de croisement plus ou moins complexes sur un problème donné mais l'implémentation de ces principes est souvent liée intrinsèquement au problème.

Remarque : Cette méthode de diversification des éléments rappelle la méthode du simplexe ou des polytopes. Dans cette méthode, les solutions des problèmes sont

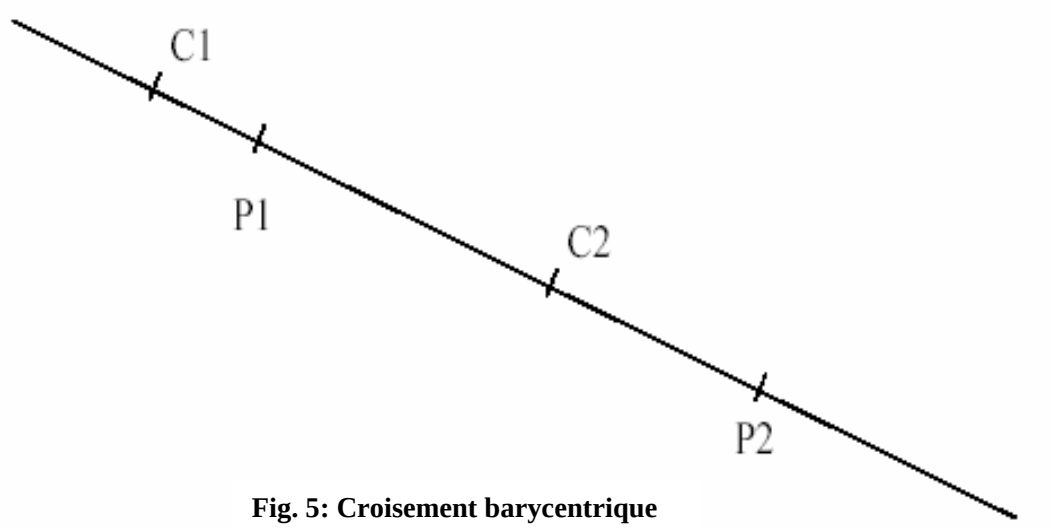


Fig. 5: Croisement barycentrique

recherchée de manière itérative en considérant une population initiale constituant des sommets de polytopes de l'espace d'état et où les éléments successifs sont construits de manière barycentriques. Nous pouvons pousser la comparaison, puisque les éléments sont réévalués après cette diversification dans les deux méthodes. Les algorithmes génétiques offrent cependant un aspect stochastique important (les éléments sont reproduits avec la probabilité P_c) et une efficacité supérieure.

3.2.6. Opérateur de mutation

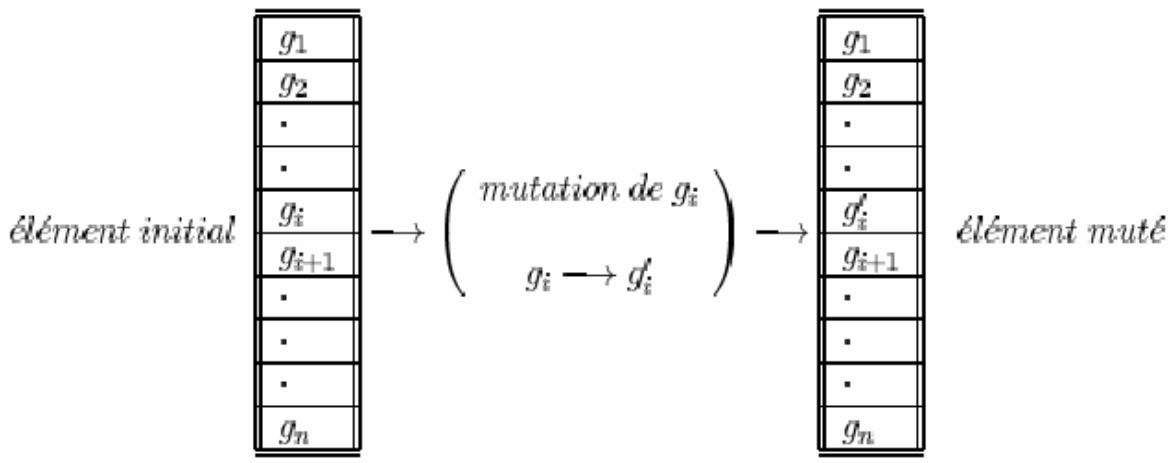
L'opérateur de mutation apporte aux algorithmes génétiques l'aléa nécessaire à une exploration efficace de l'espace. Cet opérateur nous garantit que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi, en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implantations fonctionnent de cette manière [Michalewicz ,1991] et sont conformes aux résultas théoriques de R. Cerf [Cerf ,1994], voir la quatrième section . Les propriétés de convergence des algorithmes génétiques sont donc fortement dépendantes de cet opérateur.

Pour bien comprendre l'utilité de l'opérateur de mutation, considérons un problème discret pour lequel les individus sont codés sous forme de chaînes chromosomiques constituées de trois valeurs entières et prenons une population de dix individus. On suppose de plus que le croisement utilisé est un croisement à découpage de chromosomes à un point peu importe lequel. Si par malchance, la population initiale ne présente pas de « 7 » en troisième position, par exemple:

θ_1 :	5	4	0
θ_2 :	7	3	3
θ_3 :	4	6	6
θ_4 :	2	7	4
θ_5 :	6	8	5
θ_6 :	5	5	2
θ_7 :	9	8	8
θ_8 :	4	9	9
θ_9 :	7	3	0
θ_{10} :	8	2	3

et que l'optimum se trouve en ‘4.5.7’ , il est impossible d'atteindre ce point avec l'opérateur de croisement. L'opérateur de mutation permet alors de faire varier aléatoirement la valeur des composantes de ces termes, c'est à dire des gènes.

La mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (voir ci dessous). Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur initiale. Dans le cas d'éléments codés en binaire, on remplacera la valeur ‘0’ par ‘1’ et vice-versa.



Dans les codages réels, on procède un peu de la même manière en tirant aléatoirement un élément, auquel on ajoute un bruit aléatoire en veillant à ce que l'élément résultant reste dans le domaine d'extension qui lui est propre.

Le choix pratique des probabilités P_m et P_c dépend de la complexité du problème étudié, cependant il parait souhaitable de faire dépendre ces paramètres de la génération courante, afin d'explorer plus largement l'espace, au ‘début’ de l'algorithme, et moins sur la ‘fin’.

Comme nous l'avons déjà précisé, les opérateurs de croisement et de mutation ne font pas intervenir la fonction à optimiser, ce sont des opérateurs stochastiques d'exploration du domaine admissible. C'est l'opérateur de sélection qui guide la population vers les valeurs élevées de la fonction.

4. Conclusion

Dans ce chapitre on a essayé d'introduire les algorithmes génétiques, qui représentent pour notre problème une approche pour l'évolution des contrôleurs neuronaux des agents. Les opérateurs définis sont les mêmes utilisés dans le problème de la poursuite mais à la différence des codages des gènes.

La genèse des Réseaux de Neurones Récurrents

1. Introduction

Le besoin de développement d'architectures nouvelles de réseau se faisait sentir face aux limites des réseaux à couches d'autres architectures de réseaux. La volonté d'incorporer explicitement de informations connues a priori dans l'architecture. La recherche sur la stabilité des réseaux de neurones ainsi que la vitesse de convergence des algorithmes d'apprentissage concourent aussi à cette volonté de développer de nouvelles architectures.

Parmi ces architectures on trouve une classe très générale de réseaux de neurones : les réseaux récurrents.

Les réseaux récurrents ou dynamiques ou feedback sont caractérisés par des connexions bouclées. Ils permettent alors la modélisation de système à mémoire interne ainsi que des systèmes variants dans le temps. Ainsi grâce à sa mémoire adaptable il a la capacité d'implémenter directement des systèmes dynamiques.

Ces réseaux récurrents sont caractérisés par des cycles si on les représente sous forme de graphes. Ceci introduit une dimension temporelle dans son comportement. Ils se caractérisent par deux types de comportement dynamique :

- Une dynamique autonome convergente (pour des entrées fixées)
- Une dynamique non-autonome non-convergente (pour des entrées variantes dans le temps)

Le réseau de Hopfield et la machine de Boltzman rentrent dans la première catégorie. Les réseaux de la deuxième catégorie utilisent le contexte courant et leurs entrées variantes dans le temps pour évoluer suivant leur dynamique.

2. Le Réseau de Hopfield : premier réseau récurrent

C'est le premier exemple remarquable de réseau récurrent.

En 1982, John Hopfield publie un article qui va relancer les réseaux de neurones. Selon lui le système nerveux recherche les états stables et attracteurs de son espace d'état. Les états voisins tentent de se rapprocher de ces états stables. Il propose donc un modèle nouveau. Ce modèle est basé sur un réseau de neurone type MacCulloch et Pitts avec tous les neurones connectés entre eux (mais il n'y pas d'auto-connexion). Les entrées (à l'origine binaire) sont multivaluées. La règle d'apprentissage est la règle de Hebb. L'autre apport est la comparaison avec la physique en ce qui concerne les verres de Spin : on peut constater une analogie entre la relaxation du réseau de neurones vers un état stable et les verres de Spin d'Ising, dont la fonction d'énergie décroît vers un minimum local.

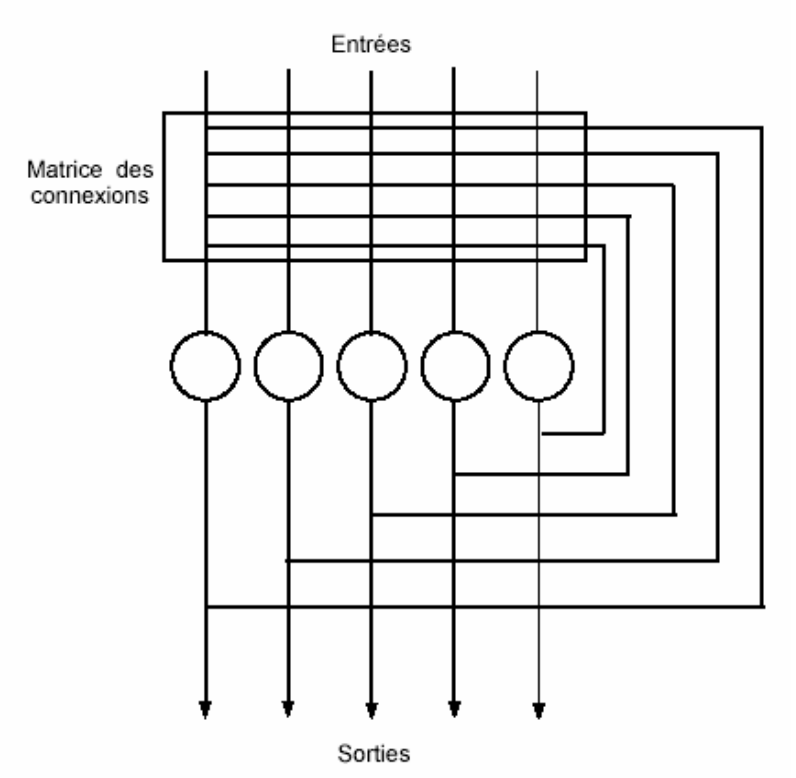


FIG. 1 –Représentation d'un réseau de Hopfield

Le réseau est une fonction dynamique supposée séquencée par une "horloge". Chaque neurone modifie son état à un instant aléatoire de façon asynchrone par rapport aux autres. Une formulation consiste à dire qu'à chaque top d'horloge on tire un neurone au hasard et on évalue son état.

Il existe deux méthodes de mise à jour

- Dynamique synchrone : tous les neurones sont mis à jour simultanément

- Dynamique asynchrone séquentielle : mise à jour l'un après l'autre dans un ordre défini.

Le modèle du neurone est le même que celui du perceptron. On constate donc que le réseau de Hopfield est récurrent et de plus il est biologiquement plausible. On peut représenter ce réseau de Hopfield par la structure suivante (Cf. figure 1:

Il y a donc rebouclage du réseau sur lui-même. Hopfield a pu exprimer une fonction de Lyapunov pour de telle structure de réseau de neurones. Cette fonction assure la stabilité du système grâce à l'égalité suivante : La fonction de Lyapunov est une entité mathématique utilisée pour l'étude de l'équilibre des systèmes complexes. Une fonction est dite de Lyapunov si elle admet un extremum pour chaque position d'équilibre du système. L'étude analytique de la fonction de Lyapunov permet de caractériser chaque position de cet équilibre (instable, asymptotiquement stable, stable, oscillatoire...).

Dans le cas de Hopfield, ce type de réseau est intéressant pour le fonctionnement dynamique, ce qui lui confère une capacité de généralisation impressionnante. Quand on lui applique de nouvelles entrées, le réseau évolue vers l'état d'équilibre le plus proche au sens de Lyapunov. L'évolution dynamique lui confère une meilleure généralisation.

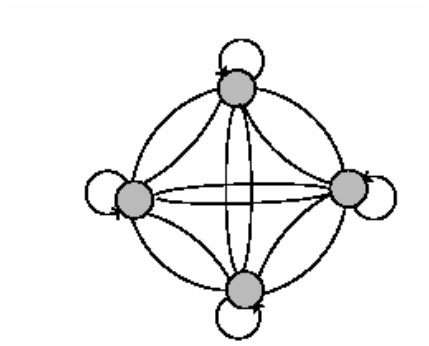


FIG. 2 – Réseau complètement interconnecté récurrent

Ce modèle est relativement différent des modèles à couches. L'apprentissage est statique mais la relaxation du réseau est dynamique c-à-d que le réseau peut effectuer un certain nombre d'itérations avant de trouver un état stable.

3. Utilisation dans les problèmes d'optimisation

3.1 Recuit simulé et réseau de Hopfield

Les états poubelles peuvent être de très fort attracteurs : Mais ils ne correspondent pas à un état désiré.

La solution consiste en une modification des poids afin d'éviter ces états poubelles. Le désapprentissage grâce à l'application de la règle de Hebb en sens inverse permet d'éviter ces états poubelles.

L'oubli catastrophique consiste en l'oubli des états attracteurs déjà appris lorsque l'on essaie d'apprendre un nouveau. A ce moment on a dépassé la capacité du réseau. (Capacité max. = rapport entre exemples à apprendre et nombre de neurones = 0.14)

Mais cette une architecture très délicate à mettre en oeuvre car l'apprentissage est beaucoup plus difficile que dans le cas du perceptron. Ceci est du au fait du nombre très grand des minima locaux engendrés par le nombre des connexions.

La machine de Boltzman est un réseau de Hopfield dans lequel l'apprentissage se fait par un algorithme de recherche aléatoire de minimum d'énergie : **Le recuit simulé.**

3.2 Les réseaux statiques récurrents ou réseaux bouclés

Ils ne sont plus nécessairement constitués en couches. L'introduction de bouclage dans leur structure rend plus difficile l'obtention d'un point d'équilibre (Cf. figure 2)

Ce type de réseau est similaire au réseau de Hopfield et il comporte plusieurs type d'adaptation de ses neurones : Adaptation asynchrone, synchrone, continue ou tous les neurones sont adaptés continuellement et simultanément.
La réponse d'un neurone i à l'équilibre est donnée par l'équation :

$$V_i = f(S_i) = f\left(\sum_{j=0}^{N} \omega_{ij} V_j\right)$$

V_i est la sortie du neurone, f est la sigmoide

Algorithme d'apprentissage utilisé

Dans le cas de la machine de Bolzman il s'agit du recuit simulé.

La rétropropagation récurrente

A la suite de la diffusion de l'algorithme de rétropropagation pour les réseaux feedforward, nombreux sont les chercheurs qui ont essayé d'explorer la possibilité d'étendre cet algorithme aux cas des réseaux avec des connexions récurrentes. Une généralisation de la rétropropagation pour les réseaux récurrents, avec séquences de relaxation où chaque neurone suit les équations suivantes :

$$\tau_i \dot{y}_{t,i} + y_{t,i} = f\left(a_{t,i} + I_{t,i}\right) \text{ avec } a_{t,i} = \sum_j \omega_{ij} y_{t,j}$$

et *f()* la fonction sigmoïde*;* y_t la sortie dans le temps t ,$a_{t,i}$ activation du neurone *i*.

Pineda a découvert que l'on pouvait appliquer directement l'algorithme de rétropropagation aux réseaux de neurone statiques récurrents. Leurs principales applications concernent l'implémentation de mémoire associative, la résolution de problème contraint.

3.3 Les Réseaux Temporels récurrents

La plupart des applications utilisent les réseaux de neurones non-bouclés (non-récurrents). Alors lorsque l'on a besoin de mémoire interne, on contourne et on utilise en entrée un registre à décalage qui se comporte comme une mémoire externe. Ainsi même des réseaux avec des connexions retardées (Time delay neural network Lang 88) le réseau FIR (réponse impulsionnelle finie) ne peuvent capturer que des informations durant un temps limité. Ils ont donc une mémoire à court terme. Cela ne leur permet pas de prendre en compte des phénomènes temporels de longues échéances.

Une attention particulière a été apportée au réseau dédié au traitement des séquences temporelles : il en résulte ce que l'on appelle les réseaux temporels récurrents. Les signaux récurrents sont retardés et permettent ainsi la mémorisation et le rappel de l'état antécédent .Ils peuvent ainsi détecter et stocker des corrélations entre des entrées successives dans le temps.

3.4 Les réseaux de Jordan et de Elman

Ils portent aussi le nom de Time Delay Récurrent Neural Networks du fait de leurs connexions retardées. Les plus connus sont les réseaux de Jordan et de Elman

a. Le réseau de Jordan

C'est une classe simple de réseaux récurrents qui utilise un vecteur d'état qui contient les copies des activations de la couche de sortie du réseau à l'instant précédent (Cf. figure

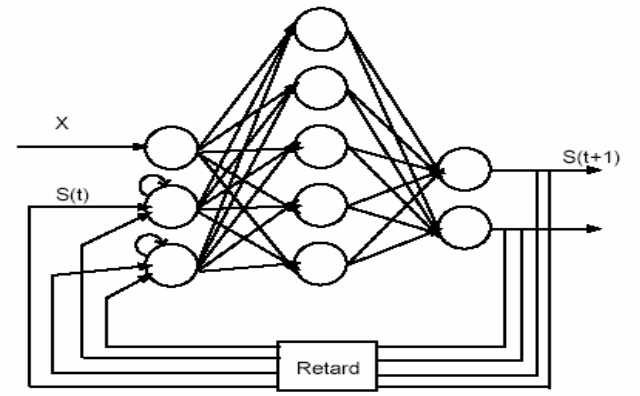


FIG. 3 –Exemple de Réseau de Jordan

3). Les connexions de ce vecteur d'état sont pondérées dans le sens Vecteur d'état →couche de sortie. Les connexions du vecteur d'état vers la couche cachée sont pondérées.

On peut comparer ce réseau à un réseau de type feedforward dans lequel on ramène les sorties du réseau avec un retard afin de générer une séquence temporelle. Il s'agit donc d'une classe réduite de réseaux récurrents. Le principal désavantage de ce type de réseau réside dans le fait que l'état qui doit être retenu par le réseau doit se manifester dans le calcul des sorties du réseau. La représentation de nouvelles structures temporelles ne peut pas être créée.

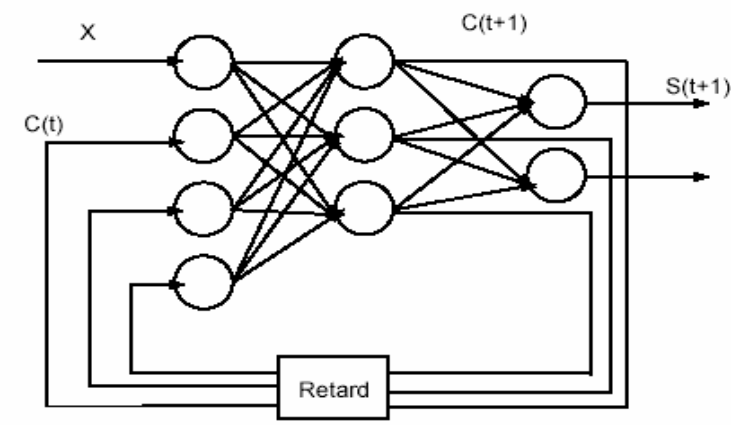


FIG 4 Exemple de réseau de Elman

L'apprentissage de tel réseau est assuré par un apprentissage supervisé.

b. Le Réseau de Elman

Ce type de réseau est basé sur la même architecture que le réseau de Jordan, mais Elman lui utilise les copies des activations des couches cachées à l'instant précédent pour le vecteur d'état.

Dans ces deux types de réseaux le vecteur d'état permet au réseau de stocker des

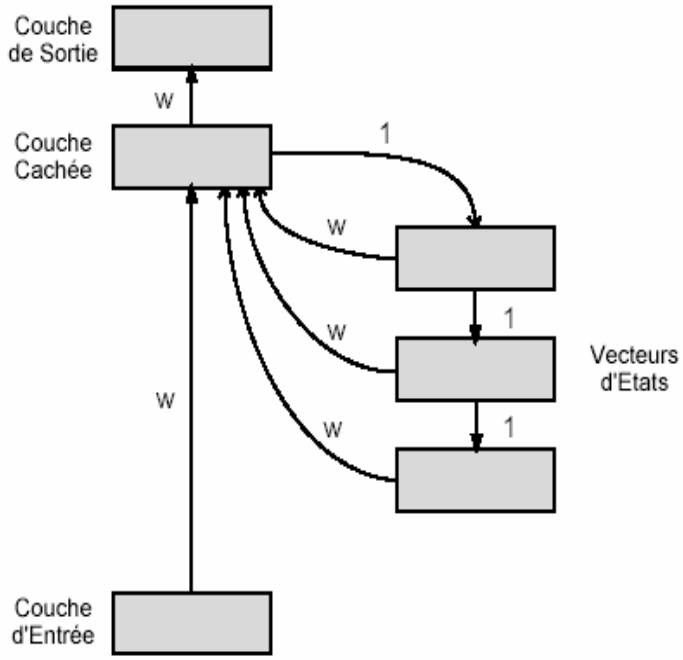


FIG 5- Exemple de tour de Elman

informations sur les entrées précédentes. C'est pourquoi ils sont utilisés dans la prédiction de séries temporelles, les tâches de classification, l'apprentissage de scénarii. Par rapport au réseau de type feedforward qui travaille avec des fenêtres temporelles, ils ont l'avantage de tenir compte implicitement du contexte temporel.

Pour l'apprentissage de son réseau, Elman a développé une version de la rétropropagation dans le temps en ne conservant qu'un ou deux coups d'échantillonnage. Cette réduction rend l'algorithme direct.

Ces réseaux sont utilisés pour l'apprentissage de structure séquentielle [Wilson 95]. Les réseaux d'Elman sont utilisés pour l'apprentissage de structure grammaticale d'un ensemble de phrases générées aléatoirement.

c. Les tours de Jordan et de Elman

Par la suite afin d'augmenter les performances de leurs réseaux, Elman et Jordan ont décidé d'intégrer plusieurs vecteurs d'états à l'intérieur de leurs réseaux : on parle alors de Tour de Jordan (Cf. figure 6) et de Tour d'Elman (Cf. figure 5). L'idée étant en multipliant les vecteurs d'états de stocker un maximum d'informations sur les états précédents. En effet il s'avère qu'avec le même nombre de poids (connexions) mais avec un nombre de vecteurs d'états plus grand le réseau de neurones apprend plus vite.

Les limites des réseaux de Jordan et Elman ont une récurrence restreinte complètement déterminée par la structure. Le gros problème de cette structure est qu'elle ne peut tenir compte des structures temporelles évolutives. En effet les retards sont fixés dans l'architecture du réseau et la part des données réutilisées en entrée est fixée par le concepteur. Ceci peut être tout à fait convenable si on a une connaissance a priori du problème à résoudre.

Problème du réseau de Jordan : son apprentissage est supervisé et d'après sa structure les récurrences apparaissent sur la couche de sortie. Si on supervise ces sorties la récurrence est orientée et on obtient en fait un réseau feedforward avec une entrée en plus

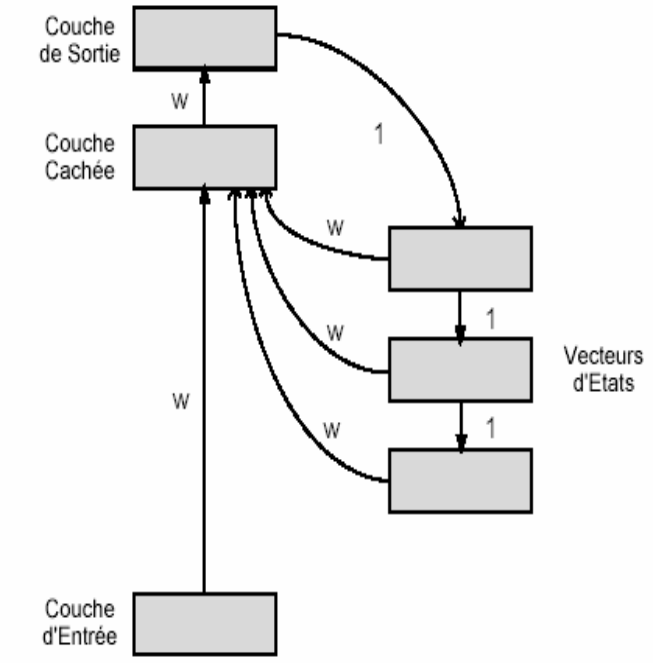


FIG. 6 exemple de tour de Jordan

Qui est la sortie du réseau retardée. C'est un TDNN (Time Delay Neural Network). Ainsi la représentation interne temporelle ne peut être représentée car le réseau ne tient compte que des sorties.

Dans les réseaux récurrents en général on passe outre cette contrainte et ils sont alors capables de capter la séquence temporelle caractéristique de la fonction à apprendre.

4. Les algorithmes d'apprentissage associés

4.1 Algorithmes d'apprentissage du point fixe

Dans ces algorithmes on suppose que le réseau converge vers des points fixes stables. Cette convergence vers des points fixes peut être très intéressante pour les problèmes contraints et les tâches de mémoires associatives. Le problème est que les

réseaux récurrents ne convergent pas tous vers ces points fixes. Mais il existe un certain nombre de cas qui garantissent une convergence :

Des conditions linéaires sur les poids garantissent que la fonction de Lyapunov suivante :

$$L = -\sum_{i,j} \omega_{ij} y_i y_j + \sum_j (y_i \log y_i + (1 - y_i) \log(1 - y_i))$$

décroît jusqu'à un point fixe.

Atiya a montré qu'un point fixe unique est atteint en fonction des conditions initiales si $\sum_{ij} \omega_{ij}^2 \leq \max(\sigma')$ D'autres études empiriques indiquent qu'appliquer des algorithmes du point fixe stabilisent les réseaux en les forçant à un comportement asymptotique. Mais il n'y a pas encore de démonstration de cette remarque.

Problème avec les algorithmes du point fixe : Même si la convergence vers un point fixe est assurée, ces algorithmes d'apprentissage peuvent rencontrer des difficultés. La procédure d'apprentissage calcule la dérivée de la somme des erreurs mesurées par rapport aux paramètres internes du réseau. Ce gradient est utilisé dans une procédure d'optimisation, typiquement une variante de la descente du gradient afin de minimiser une erreur De telles procédures d'optimisation suppose que la surface d'erreur générée par les différents calculs est continue, et donc si cette supposition n'est pas vérifiée alors l'algorithme échoue.

4.2 La Rétropropagation récurrente

Pineda et Almeda ont découvert que l'algorithme de rétropropagation de l'erreur est un cas particulier d'une procédure de calcul du gradient de l'erreur plus général. Les équations sont les suivantes:

$$y_i = \sigma(x_i) + I_i \qquad (1.1)$$

$$z_i = \sigma(x_i) \sum_j \omega_{ij} z_i + e_i \qquad (1.2)$$

$$\frac{\partial E}{\partial \omega_{ij}} = y_i z_i \qquad (1.3)$$

où z_i est la dérivée partielle de E par rapport à y_i, E est une erreur métrique sur $y(t_\infty)$ et $e_i = \frac{\partial E}{\partial y_i(t_\infty)}$ est la dérivée première de E par rapport à l'état final d'un neurone.

Une solution pour calculer un point fixe est de calculer: $0 = -y_i + \sigma(x_i) + I_i$ et le point fixe sera atteint lorsque $\frac{dy_i}{dt} = 0 d'où k \frac{dy_i}{dt} = -y_i + \sigma(x_i) + I_i$. En considérant l'équation 1.2 on obtient une solution pour $\frac{dz_i}{dt} = -z_i + \sigma'(x_i)\sum_j \omega_{ij} z_j + e_i$

Ces équations peuvent être implémentées directement en supposant que le temps passe au calcul est négligeable par rapport au temps passé au point fixe et que le taux de changement de poids η est lent comparé à la vitesse de présentation des échantillons d'apprentissage. Les poids seront donc mis à jour par une équation du type $\frac{d\omega_{ij}}{dt} = -\eta \frac{\partial E}{\partial \omega_{ij}} = -\eta y_i z_j$ où si on a introduit un terme de biais $0 \le \alpha \le 1 on a \frac{d^2\omega_{ij}}{dt^2} + (1-\alpha)\frac{d\omega_{ij}}{dt} + \eta y_i z_j = 0$

4.3 La rétropropagation dans le temps ou dépliage dans le temps

Les algorithmes d'apprentissage du point fixe tels la rétropropagation du gradient de l'erreur sont incapables de capter les attracteurs non fixes ou de produire un comportement temporel désiré sur un intervalle borné. Il faut donc adapté les procédures pour ces situations de points non fixes. On doit donc minimiser une erreur de trajectoire de fonction E(y) entre t0 et t1.Donc $E = \int_{t_0}^{t_1} (y_0(t) f(t))^2 dt$ mesure la déviation de y_0 par rapport à f. En minimisant E on tend à rapprocher y_0 de f. On va donc utiliser une technique de calcul de $\frac{\partial E(y)}{\partial \omega_{ij}}$ dérivée qui nous permettra d'utiliser la descente de gradient afin de minimiser E.

Afin de mener à bien cette opération il faut conserver en mémoire toutes les activations des neurones au cours du temps dans la passe de propagation avant, afin de pouvoir utiliser l'algorithme de rétropropagation du gradient dans la phase propagation arrière.

Dans cette approche simple on opère un dépliage temporel du réseau continu en un réseau échantillonné avec une valeur d'échantillonnage Δt et on applique la rétropropagation usuelle sur ce réseau discrétisé. En prenant assez petit on conserve un apprentissage quasi-continu dans le temps. Pratiquement on va donc dupliquer les neurones T fois pour une séquence temporelle T. On obtiendra ainsi un réseau déplié dans le temps qui duplique les connexions au cours du temps ce qui lui permet de garder en mémoire l'évolution. Ce réseau résultant possède une structure multicouche c'est pourquoi on peut utiliser la rétropropagation comme dans un réseau non bouclé.

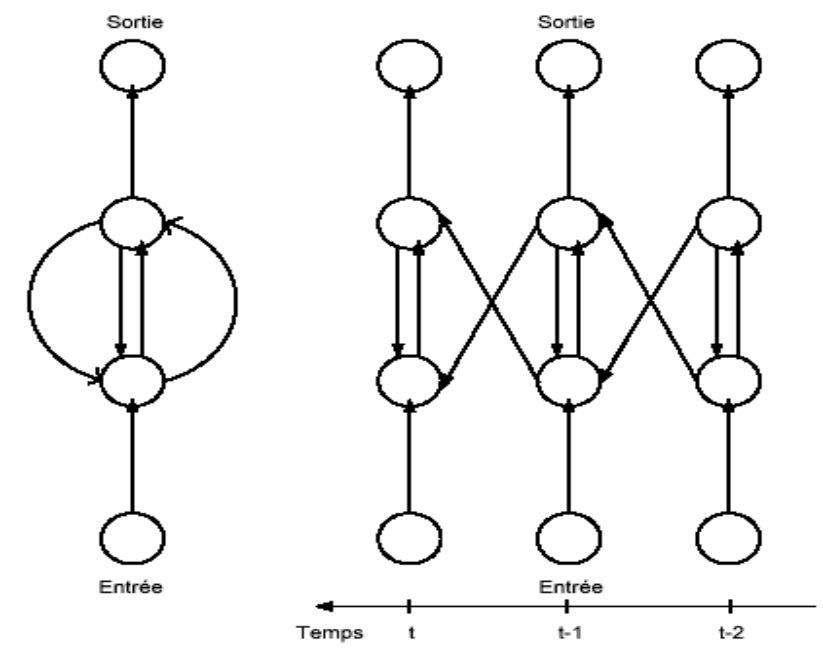


FIG. 1.7 – Dépliage temporel d'un réseau récurrent simple

Cet algorithme est local dans l'espace (les calculs de gradient sont locaux sur un échantillon E/S) mais il n'est pas local dans le temps dès que l'on doit stocker toutes les valeurs passées des activations. Le plus gros problème dans l'implémentation d'un tel algorithme est les ressources qu'il exige. Le complexité spatiale et surtout temporelle qui résulte de la duplication des neurones (Cf. figure 7) en fait un algorithme quasi-inutilisable. Dès que les séquences temporelles deviennent longues il devient impossible à utiliser du fait de l'explosion combinatoire.

4.4 Apprentissage temporel récurrent en temps réel (Real Time Recurrent Learning RTRL)

Développé par Williams et Zipser C'est un autre algorithme pour l'apprentissage des réseaux de neurones récurrents sans restriction sur l'architecture, qui évite la rétropropagation dans le temps. Il opère de façon récursive en gardant en mémoire pendant la passe de propagation avant les dérivées partielles qui indiquent comment évolue chaque poids du réseau.

La mise à jour des poids se fait en ligne à chaque itération sans attendre la fin de la séquence. Il n'y donc plus de contrainte sur la durée de la séquence et donc la complexité temporelle est indépendante de la durée de la séquence. Afin de pouvoir faire cette adaptation en ligne à chaque couple E/S, on doit calculer le gradient sur la fonction d'erreur mais en local par rapport au paramètre du système $\{\omega_{ij}\}$ et le stocker.

Par linéarité le gradient se décompose dans le temps comme la somme des gradients à chaque instant soit $\Delta\omega_{ij}(t) = -\eta\frac{\partial E(t)}{\partial \omega_{ij}} = \eta\sum_k E_k(t)\frac{\partial y_k(t)}{\partial \omega_{ij}}$ On obtient ainsi une équation itérative en $\frac{\partial y_k}{\partial \omega_{ij}}$ qui permet d'actualiser les dérivées partielles.

C'est un algorithme local dans le temps mais non local dans l'espace du fait du calcul du gradient $\frac{\partial y_k}{\partial \omega_{ij}}$. Cet algorithme souffre d'une complexité excessive et le nombre d'opérations requises dans le pire des cas est de l'ordre de $O(N^4)$(N est le nombre d'unité du réseau). D'après Bengio et Tsoi cette architecture souffre d'un manque de stabilité et converge difficilement vers une solution optimale.

a. Problème théorique avec la descente de gradient

Dans la plupart des problèmes les réseaux de neurones récurrents sont bien meilleurs que les réseaux de types feedforward. Mais ils sont aussi plus difficiles à entraîner. Des expériences montrent qu'ils stagnent souvent dans des solutions sous-optimales qui tiennent compte des dépendances à cours termes et non à long terme. D'après Mozer la rétropropagation n'est pas assez efficace pour apprendre de longues séquences temporelles. Une remarque importante au niveau de l'algorithme de rétropropagation est qu'il perd de sa mémoire les événements les plus anciens et qu'il tient compte en grande partie des événements les plus récents (on parle d'évanouissement du gradient). Ceci est du principalement au calcul du gradient.

On peut alors définir trois choses indispensables pour réaliser un bon apprentissage et stockage de séquences temporelles :

- Que le système soit capable de stocker des séquences temporelles de durée arbitraire
- Le système doit être robuste (au bruit)
- Que les temps d'apprentissage ne soit pas trop longs

b. Nouveaux algorithmes pour l'apprentissage des réseaux récurrents

Nous venons de voir que l'un des principaux problèmes pour l'apprentissage des RNR est le stockage en mémoire interne de séquences temporelles longues. La rétropropagation semble alors une méthode inadéquate.

L'une des solutions afin de contourner le problème est d'utiliser la connaissance a priori du système.

c. Optimisation tempo-pondérée de pseudo-newton

L'algorithme de pseudo-newton à l'avantage de recaler le pas d'apprentissage de chaque poids dynamiquement. L'intérêt de l'ajustement du pas d'apprentissage serait peut-être de résoudre le problème du gradient qui disparaît (pour les événements anciens).

L'algorithme de Pseudo-Newton calcul une approximation diagonale de la matrice Hessienne (dérivée seconde de la fonction de coût par rapport aux paramètres) et ajuste les paramètres d'après la règle suivante: $\Delta\omega_i(p) = -\frac{\eta}{\frac{\partial^2 C(p)}{\left|\partial^2 \omega_{ij}\right|} + \mu} \times \frac{\partial C(p)}{\partial \omega_i}$ ou $\Delta\omega_i(p)$ est la mise à jour pour le poids ω_i après que l'échantillon aie été présenté au réseau et que le coût C(p) aie été calculé. η *et* μ sont de petites constantes positives. L'algorithme de pseudo-Newton tempo-pondéré est basé sur l'idée de considérer le dépliage temporel du réseau récurrent et chaque instance d'un poids (à chaque t) comme une variable. Le mise à jour des poids se fait alors d'après l'équation suivante: $\Delta\omega_{it}(p) = -\frac{\eta}{\frac{\partial^2 C(p)}{\left|\partial^2 \omega_{ij}\right|} + \mu} \times \frac{\partial C(p)}{\partial \omega_{it}}$ est une instance pour chaque temps t de ω_i. Il permet de meilleures performances si on adapte en ligne μ avec un "momentum".

d. Propagation de l'erreur discretisée

La base du problème pour l'algorithme de rétropropagation vient du fait du stockage de l'information. Le gradient rétropropagé disparaît quand le système est dans un état stable pendant un moment. Afin de bien opérer il faudrait se souvenir de l'erreur au moment ou le système est entré dans cet état stable.

L'idée est donc de développer un algorithme qui propage l'erreur discrétisée à travers les neurones à fonction non dérivable (seuil) du réseau. Le réseau peut être considéré comme une série d'éléments locaux avec chacun une fonction de propagation avant et une fonction d'erreur.

Il en résulte que lorsqu'un élément avec une fonction d'activation non dérivable est auto-bouclé avec un gain > 1 deux points stables sont induits et le pseudo gradient ne disparaît pas avec le temps.

5. Le réseau de neurones Aléatoires

Ce type de réseaux de neurones statiques récurrents introduit par Galenbe en 1989 est différent de l'approche de MacCulloch & Pitts. Ce réseau est conçu à partir d'un nombre d'unités stochastiques interconnectées entre lesquelles circulent des signaux soit excitateurs (> 0) soit inhibiteurs (< 0).

Chaque unité est associée à un compteur $k_i(t)$ son potentiel à l'instant t qui est incrémenté si l'unité reçoit un signal > 0 décrémenté sinon. Chaque unité évolue au gré des signaux reçus avant d'être remise à zéro lorsqu'elle passe en phase d'émission. La période aléatoire qui sépare l'émission de deux signaux suit une loi stochastique exponentielle. En plus des signaux internes, des signaux externes sont injectés dans le réseau selon le processus de Poisson.

6. Conclusion

Il y a eu un engouement pour les réseaux de neurones récurrents afin de palier le problème d'apprentissage de séquences temporelles. Un réseau connexionniste dédié aux séquences temporelles doit avoir une représentation interne du passé. C'est dans cette optique et par soucis de concilier complexité et efficacité que différents chercheurs ont développé des modèles d'architectures nouvelles.

Les réseaux dynamiques récurrents

1. Introduction

Ce chapitre va introduire le type le plus important de réseaux de neurones pour les contrôleurs neuronaux. Ce sont les réseaux dynamiques récurrents, leurs propriétés, puissance d'expression, évolution et quelques utilisations. Ce chapitre va servir à la compréhension du modèle de réseaux de neurones du chapitre de la poursuite de Cliff&Miller [Cliff].

Le "Dynamic Recurrent Network" a été introduit dans le but de donner un modèle qui avec un apprentissage raisonnable présente des caractéristiques intéressantes au niveau du traitement des séquences temporelles. Cette classe de réseaux récurrents englobe un grand nombre de réseaux de neurones introduit pour le traitement temporel

Ce modèle de réseau est basé sur des unités type MacCulloch et Pitts à fonction d'activation sigmoïdale. Son architecture est des plus générales puisqu'elle permet des connexions retardées et bouclées. Le principal intérêt de ces réseaux est sa mémoire adaptable qui lui permet d'implémenter directement un système dynamique - très important pour la dynamique des robots-. Les bouclages et les retards entre les neurones permettent de conserver sous forme codée la mémoire. Ce réseau est décrit par un système d'équations différentielles dont les variables internes sont les activations des neurones cachés dont la forme est récurrente spatialement et temporellement.

Les différents algorithmes utilisés pour l'apprentissage et la mise à jour des ses paramètres sont d'une part la propagation en chaînage avant, la propagation en chaînage arrière, un algorithme itératif en chaînage arrière (TRBP) « Time Recurrent Back Propagation ».

2. Le DRNN comme représentation de la mémoire

Afin de pallier les limitations de réseaux non-bouclés, il faut introduire explicitement une ou des récurrences dans l'architecture du réseau. Les informations intéressantes contenues dans les taches temporelles nécessitent un stockage. Mais quelques fois ces données pertinentes s'étendent sur des intervalles de temps important, ce

qui exclut la technique des fenêtres temporelles. Le réseau doit donc être muni d'une mémoire interne s'il l'on ne peut lui fournir les informations d'une manière externe.

C'est dans ce but que les poids synaptiques du réseau de type Rumhelhart sont remplacés par des fonctions de transfert d'un filtre linéaire à réponse impulsationnelle finie (FIR). (a une impulsion de durée finie correspond une réponse de durée finie). Alors l'activation $a_i(t)$ d'un neurone *i* à l'instant *t* est donnée par:

$$x_i(n) = f\left(\sum_j \sum_{d=0}^{D} \omega_{ji}^d x_j(n-d)\right) = f\left(\sum_j \omega_{ji} * x_j(n)\right)$$

où

$$\omega_{ij} = [\omega_{ji}^0, \omega_{ji}^1, \cdots, \omega_{ji}^D,]^T \text{ et } x_j(n) = [x_j(n), x_j(n-1), \cdots, x_j(n-D),]^T$$

Les coefficients du filtre doivent être adaptés afin de correspondre au mieux avec le problème à résoudre. Ceci permet aux interactions neuronales de présenter un vaste choix de mémoires formelles.

$\omega_k^d(i,j)$	poids de la connexion retardée de d entre le neurone i et le neurone j à l'instant k
$w_k^d(j)$	$= [\omega_k^d(1,j), \cdots, \omega_k^d(N,j)]^T$
$W_k^d(j)$	$= [w_k^d(1,j), w_k^d(2,j), \cdots, w_k^d(N,j)]^T$
s_k	$= \sum_{d=0}^{D} W_k^{dT} v_{k-d}$
v_k	$= g(s_k)$ le vecteur d'activation des neurones
x_k	vecteur d'entrée du réseau à l'instant t

Tab. 1.2 - Résumé les notations d'un DRNN

Cette méthode des connexions FIR a été introduite par Wan aux réseaux à couches. On trouve ici son extension aux réseaux bouclés. La prédiction apparaît comme le domaine de prédilection de ce type de réseau car sa version non bouclée est utilisée dans la prédiction à court terme.

3. Une dynamique à relaxation

Le DRNN est un réseau totalement bouclé de N neurones. ω_{ij}^d représente le poids de la connexion synaptique retardée de *d* unités de temps entre le neurone *i* et le neurone *j*. D représente le délai maximal c'est à dire l'ordre du filtre. L'activation est la réponse d'un neurone et $v_k = (v_k \in \Re^{[N+1]})$ sont les vecteurs d'activités des neurones à l'instant *k.* le réseau possède des neurones d'entrées dont l'activation est $x_k = (x_k \in \Re^{[N+1]})$. Les valeurs désirées des neurones de sorties sont notées $d_k = (d_k \in \Re^{[N+1]})$. Le vecteur poids retardé de

d unités de temps est noté $\omega_k^d(j) = \left[\omega_k^d(1,j), \omega_k^d(2,j), ..., \omega_k^d(N,j)\right]^T$ et la matrice de dimension NxN $W_k^d(j) = \left[\omega_k^d(1), \omega_k^d(2), ..., \omega_k^d(N)\right]^T$ contient tous les poids des connexions retardés d'un délai $d < D$.

On détermine les paramètres du système à chaque étape k en relaxant le système

$$T\frac{dv_k(t)}{dt} = -v_k(t) + g(s_k(t)) + x_k$$

$$s_k(t) = \sum_{d=1}^{D} W_k^{d\,T} v_{k-d} + W_k^{0\,T} v_k(t)$$

avec le vecteur T,$T = [\tau_1; \tau_2; \quad ; \tau_n]$ convenablement choisi pour amener dans un temps raisonnable le système vers un état d'équilibre.

s_k désigne le vecteur des entrées internes du neurone à l'itération k, et g() est une sigmoïde de la forme $g(s_k) = [g(s^1_k); \_ \_ \_ ; g(s^n_k)]T$ avec $g(s) = \frac{1}{1-\exp^{-\beta s}}$ où β est la pente de la sigmoïde.

La convergence d'un tel système vers un point fixe n'est pas garantie (Hertz 91). On suppose donc qu'il en existe au moins un. Si le système admet donc un point d'équilibre stable ou instable il verifie $v_k = g(\sum_{d=0}^{D} W_k^{dT} v_{k-d}) + x_k$

On doit remarquer que le système doit garder en mémoire la trace des D dernières actions, ce qui conduit à un volume de stockage de DN valeurs. On peut résumer les notations d'un *DRNN* dans le tableau 1.2

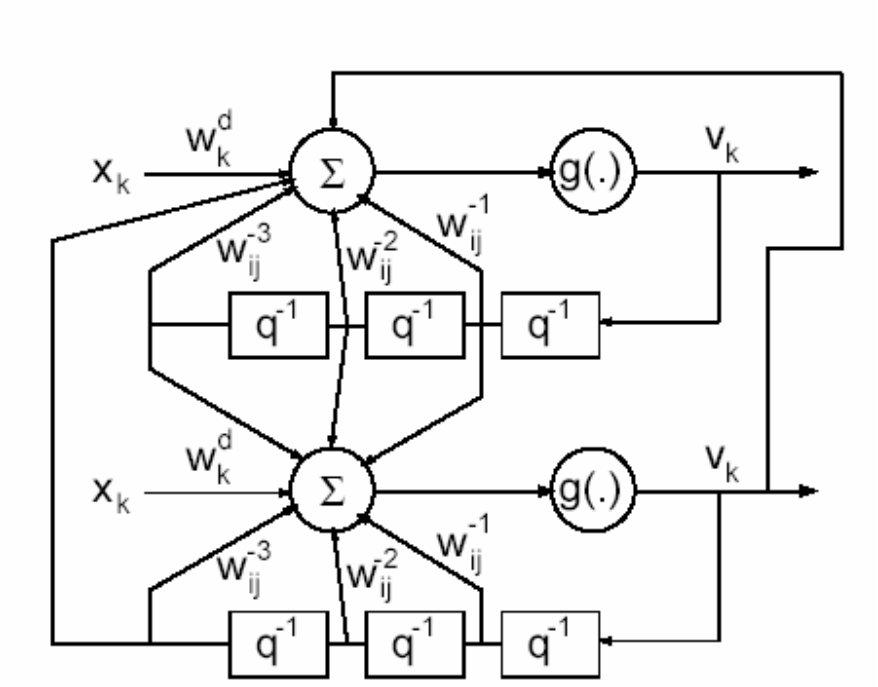


Fig. 1 Exemple de DRNN simple avec deux entrées et deux sorties

4. Les Algorithmes d'apprentissage des réseaux dynamiques récurrents

4.1. Méthodes d'apprentissage

Afin d'acquérir sa "connaissance", un réseau de neurones passe par une phase d'apprentissage. Celle-ci s'effectue grâce à un algorithme d'apprentissage. Il existe beaucoup d'algorithmes d'apprentissage et le choix d'un d'entre eux reste difficile. Pour un même problème leur efficacité sera différente. Il faut donc choisir l'algorithme d'apprentissage en fonction du problème. Une des méthodes d'apprentissage des réseaux de neurones est la rétropropagation du gradient de l'erreur.

4.1.1. La rétropropagation du gradient de l'erreur

L'algorithme de rétropropagation du gradient de l'erreur à été crée en généralisant les règles d'apprentissage de Widrow-Ho_ aux réseaux multicouches à fonction de transfert non-linéaire. C'est un algorithme utilisé avec des réseaux de type feedforward pour l'apprentissage de fonction, la reconnaissance de formes et la classification.

a. Principe

La rétropropagation du gradient de l'erreur est utilisée pour ajuster les poids et les biais du réseau afin de minimiser l'erreur quadratique entre la sortie du réseau et la sortie réelle. A chaque couple entrée/sortie, une erreur est calculée, le gradient, ou pente, de l'erreur est déterminé. Ensuite les poids et les biais sont modifiés en ligne sur le réseau.On réitère ces calculs jusqu'à l'obtention du critère d'arrêt.

b. Algorithme

1. Initialisation des poids *W[q]* à des petites valeurs aléatoires
2. Présentation d'une entrée x_k et de la sortie désirée d_k
3. Calcul de la sortie actuelle par propagation à travers les couches par $y_j^{[q]} = F(\sum_i W_{ji}^q \bullet y_i^{[q-1]})$ où F est la fonction de transfert du neurone et[q] la q^e couche du réseau
4. Accumulation des erreurs en sortie $\varepsilon = \sum_k (d_k - y_k^{[s]})$ *où* d_k est la sortie désirée associée au vecteur d'entrée x_k. Où $y^{[s]}_k$ est la sortie obtenue sur la dernière couche au temps t, ε est l'erreur cumulée pour k présentations de couples (x_k,d_k).
5. Rétropropagation du gradient de l'erreur (δ) depuis la dernière couche vers la première couche
   - F Pour chaque cellule de sortie $\delta_i^{[s]} = -\left(d_i - y_i^{[s]}\right) \bullet F(p_i^{[s]})$
   - F Pour chaque cellule cachée $\delta_i^{[q]} = -\sum_k \delta_i^{[q+1]}.W_{ki}.F(\delta_i^{[p]})$

6. Mise à jour des poids selon la règle $\Delta W_{ij}^{[q]} = \alpha.(\delta_k^{[q]}.x_j^q) ou \alpha$ est le coefficient d'apprentissage compris dans l'intervalle [0; 1].
7. Retour à 2 tant qu'il y a des couples à présenter.

c. Choix du critère à minimiser

Dans le cas de la rétropropagation de l'erreur, le critère à minimiser est une erreur quadratique. L'application de l'algorithme du gradient nécessité la dérivabilité de la fonction de transfert. Le critère de minimisation d'erreur est le suivant $\varepsilon = \sum_k (d_k - y_k^{[s]})$

d. Avantages et inconvénients

d.1. Avantages Ce fût un des premiers algorithmes développés pour l'apprentissage des réseaux de neurones multicouches de type feedforward. Il permet de pallier une carence de l'algorithme du perceptron qui est incapable de modifier les poids des couches cachées. L'implémentation informatique ne présente pas de difficulté. Il a surtout permis de sortir le connexionnisme de l'impasse que représente l'apprentissage des réseaux de types feedforward multicouches en permettant la modification des poids des couches cachées.

d.2. Inconvénients En ce qui concerne l'algorithme

- L'algorithme de rétro-propagation du gradient de l'erreur suit la descente du gradient de l'erreur : un minimum local peut rapidement bloquer la recherche des optima globaux.
- L'algorithme de rétro-propagation est gourmand en temps de calcul.
- Importance du choix du coefficient d'apprentissage, si le coefficient est trop grand la dynamique du réseau va osciller autour de l'optimum, si il est trop petit, la convergence est lente et elle va se faire piéger dans un minimum local.
- Les fonctions de transfert doivent être dérivables.

4.1.2. Propagation récurrente avant et arrière

a. Introduction

L'objet de l'apprentissage est d'ajuster la matrice de poids du réseau $\{\omega^d_{ij}\}$ de façon à le faire évoluer sous l'action des entrées, d'un état initial vers des points fixes désirés correspondant aux sorties désirées du réseau. Il faudra encore minimiser une erreur quadratique. Grâce aux lois d'expansion des dérivées partielles (due à l'architecture du réseau) on obtient deux algorithmes différents : algorithme d'adaptation récursif et algorithme d'adaptation à balayage arrière ou propagation récurrente avant et récurrente arrière (reflet de la direction des dérivées partielles par rapport au données

Loi d'expansion des dérivées partielles *Soit* $z_i = f_i(z_1 \ldots . ; z_{i-1}; z_i)$

et la dérivée partielle $\frac{\partial^+ z_j}{\partial z_i} = \left[\frac{\partial z_j}{\partial z_i}\right]_{\{z_1,\ldots,z_{i-1}\}}$

Première méthode

$$\frac{\partial^+ z_j}{\partial z_i} = \sum_{k=i+1}^{j} \frac{\partial^+ z_j}{\partial z_k} \left(I - \frac{\partial f_k(z)}{\partial z_k}\right)^{-1} \frac{\partial f_k(z)}{\partial z_i}$$

Deuxième méthode

$$\frac{\partial^+ z_j}{\partial z_i} = \sum_{k=i}^{j-1} \left(I - \frac{\partial f_j(z)}{\partial z_j}\right)^{-1} \frac{\partial f_j(z)}{\partial z_k} \frac{\partial^+ z_k}{\partial z_i}$$

La propagation récurrente avant : On utilise la seconde loi d'expansion. On développe $\frac{\partial^+ E}{\partial \omega}$ *où* ω est un scalaire de la matrice W. Cet algorithme généralise le RTRL (Real Time Recurrent Learning) développé par Williams & Zipser.

La propagation récurrente arrière Propagation récurrente arrière rétropropagation arrière temporelle (TRBP). Le problème de cet algorithme vient de l'explosion numérique qu'il occasionne. (inversion matricielle, mémoire de stockage requise). D'où l'idée d'appliquer des stratégies afin de réduire la complexité en gardant à l'esprit l'efficacité de l'algorithme

- Ignorer l'influence de l'état courant sur les erreurs futures (problème de résolution des problèmes temporels).
- Solution qui consiste à approximer les dérivées partielles afin de réduire le coût de stockage.

b. La propagation récurrente arrière

Ici on utilise la première loi et ceci conduit à un algorithme sans inversion matricielle mais qui nécessite la relaxation d'un réseau adjoint (rétropropagation temporelle récurrente). L'algorithme définitif englobe la rétropropagation temporelle dans les réseaux FIR non-bouclés (Wang), pour les réseaux statiques récurrents (almeida), pour les réseaux à délais (Rumelhart). Cet algorithme peut se concevoir comme la rétropropagation récurrente appliquée à un réseau adjoint (Baldi) ou transposé (Hertz).

Un DRNN simple et son transposé sont décris par :

- inverser les signaux tels que les sorties deviennent les entrées.
- redéfinir les neurones comme des opérateurs affines en utilisant la dérivée du point fixe.
- inverser le temps.

Cet algorithme nécessite le dépliage d'un adjoint dans le temps. Cette duplication est effectuée jusqu'à ce que tous les délais aient disparu. L'algorithme produit un réseau de propagation temporelle et récurrent dans l'espace. (connexions bouclées mais non retardées uniquement). Mais ceci demande une grande capacité de stockage et de calcul :Ceci donnera l'algorithme TRBP Temporal Recurrent BackPropagation. A un instant donné les gradients sont propagés dans le réseau adjoint en temps inverse (t = k → t = 0).Le dépliage du réseau adjoint s'opère à la fin de l'itération pour permettre le calcul au fur et à mesure que les nouveaux exemples sont présentés.

c. Stabilité de la procédure d'adaptation des poids

La stabilité asymptotique de l'adaptation de la matrice de poids quand le nombre d'itérations tend vers l'infini est indispensable pour assurer le bon fonctionnement du réseau et de son apprentissage.

On peut prouver la convergence locale (sur une itération dans le réseau adjoint), les gradients sont majorés par une suite décroissante à vitesse exponentielle.

d. Critère de dépliage

On cherche à connaître le nombre de rétropropagations requises par l'algorithme afin de minimiser la mémoire requise au stockage de ces gradients.

4.1.3. L'algorithme du TRBP Temporal Recurrent BackPropagation

Cet algorithme tire avantage de la décroissance exponentielle des gradients afin de minimiser l'explosion combinatoire. L'utilisation de l'adjoint permet une symétrie utilisée par une implémentation temporelle. Sa principale caractéristique vient de l'ajustement optimal du nombre de passe dans l'adjoint afin de réduire le nombre d'opérations nécessaires.

a. Apprentissage forcé

Il suffit de remplacer les sorties calculées par le réseau par les valeurs désirées en sorties. Ressources requises : L'ordre du réseau est D la complexité spatiale est en *O(DN)*, en contraste avec O(DN3) pour la propagation récurrente avant. Une inversion matricielle n'est plus nécessaire.

b. L'évanouissement du gradient dans l'adjoint

Problème de l'apprentissage à long terme pour la propagation du gradient dans l'adjoint. Un bon apprentissage peut se résumer à deux choses :

- Le système doit être résistant au bruit.
- Le système converge en temps raisonnable.

Le gradient prend plus en compte les dépendances temporelles à court terme que les dépendances à long terme. Si le gradient tend à devenir faible sa propagation ne va pas modifier de façon suffisante les poids et va s'évanouir dans l'adjoint. C'est pourquoi cet algorithme est inadapté pour capturer les séquences temporelles à longs termes.

4.2 Le Recuit Simulé

C'est une méthode méta-heuristique destinée à résoudre les problèmes d'optimisation difficiles. Cette méthode permet en principe d'éviter les pièges des minima locaux. Elle est inspirée par la technique expérimentale du "recuit" utilisée par les métallurgistes pour obtenir un état solide bien ordonné d'énergie minimale.

a. Caractéristiques du recuit simulé

Comme d'autres méthodes récentes (Algorithmes génétiques, méthode de recherche Tabou) elle doit permettre de résoudre au mieux des problèmes difficiles composés principalement des problèmes NP difficiles (optimisation combinatoire).

Le recuit est une méthode méta-heuristique qui possède les caractéristiques suivantes

- Elle est stochastique (seule méthode praticable face à l'explosion combinatoire)
- Elle ne peuvt s'appliquer aux problèmes continus qu'après une adaptation (elles viennent du combinatoire)
- C'est une méthode directe (par opposition au gradient que l'on rétropropage)

b. La méthode du recuit simulé

Elle permet en principe de déterminer les minima globaux d'une fonction de coût. Elle est basée sur des phénomènes physiques que l'on nomme Verre de Spin. Ce phénomène

Problème d'optimisation	Système physique
Fonction objectif	Énergie libre
Paramètres du problème	"Coordonnées" des particules
Trouver une bonne configuration	Trouver des états de basses énergie

Tab. 1.3 - Analogie entre la Physique et l'Optimisation

caractérise les configurations de basse énergie de matériaux magnétiques désordonnés. La recherche des énergies minimales pose un problème d'optimisation redoutable en regard de la surface d'énergie qui présente de grandes irrégularités.

C'est pourquoi S. Kirkpatrick & al. ont proposé de régler ce problème en s'inspirant de la technique expérimentale du "recuit".

L'analogie entre la physique et le problème d'optimisation permet d'utiliser le recuit comme une méthode d'optimisation (Cf. tableau 1.3).

Pour modifier l'état d'un matériau, le physicien dispose d'un paramètre de commande : la température. Le recuit est une stratégie de contrôle de la température. En refroidissant lentement le matériau en marquant des paliers de longueur suffisante on obtient un état solide stable correspondant à un minimum d'énergie. Cette idée a conduit les chercheurs à utiliser cette méthode pour l'optimisation de problème.

c. Algorithme du recuit simulé

Basé sur deux résultats de la physique statistique qui sont :

1. Lorsqu'un équilibre thermodynamique est atteint à une température T, la probabilité de posséder une énergie donnée E est proportionnelle au facteur de Boltzman exp (ΔfracEk_BT) où k_b désigne la constante de Boltzman.
2. Afin de simuler l'évolution d'un système physique vers son équilibre thermodynamique, à une température donnée on utilise l'algorithme de Métropolis. Partant d'une configuration donnée, on fait subir au système une modification élémentaire.

Si cette transformation diminue la fonction d'énergie, alors on accepte cette transformation sinon on accepte cette mauvaise modification d'énergie ΔE avec la probabilité exp (ΔfracΔET).

En appliquant cette règle de façon itérative, on engendre une séquence de configurations qui constitue une chaîne de Markov (une configuration dépend de celle directement précédente). En principe la chaîne est de longueur infinie, mais en pratique on arrête d'appliquer la règle quand le système a atteint un équilibre thermodynamique intéressant.

On comprend alors le sens de la température T : à haute température on accepte quasiment la plupart de mouvements dans le sens opposé de la minimisation de la fonction d'énergie. L'algorithme équivaut à une simple recherche aléatoire dans l'espace des configurations.

Par contre à basse température exp (ΔfracΔET) est proche de 0. La plupart des déplacements augmentant : l'algorithme se ramène à une amélioration itérative classique.

d. Convergence théorique du recuit simulé

Sous certaines conditions, le recuit simulé converge en probabilité vers un optimum global. Ceci est intéressant par rapport à d'autres méthodes dont la convergence n'est pas prouvée.

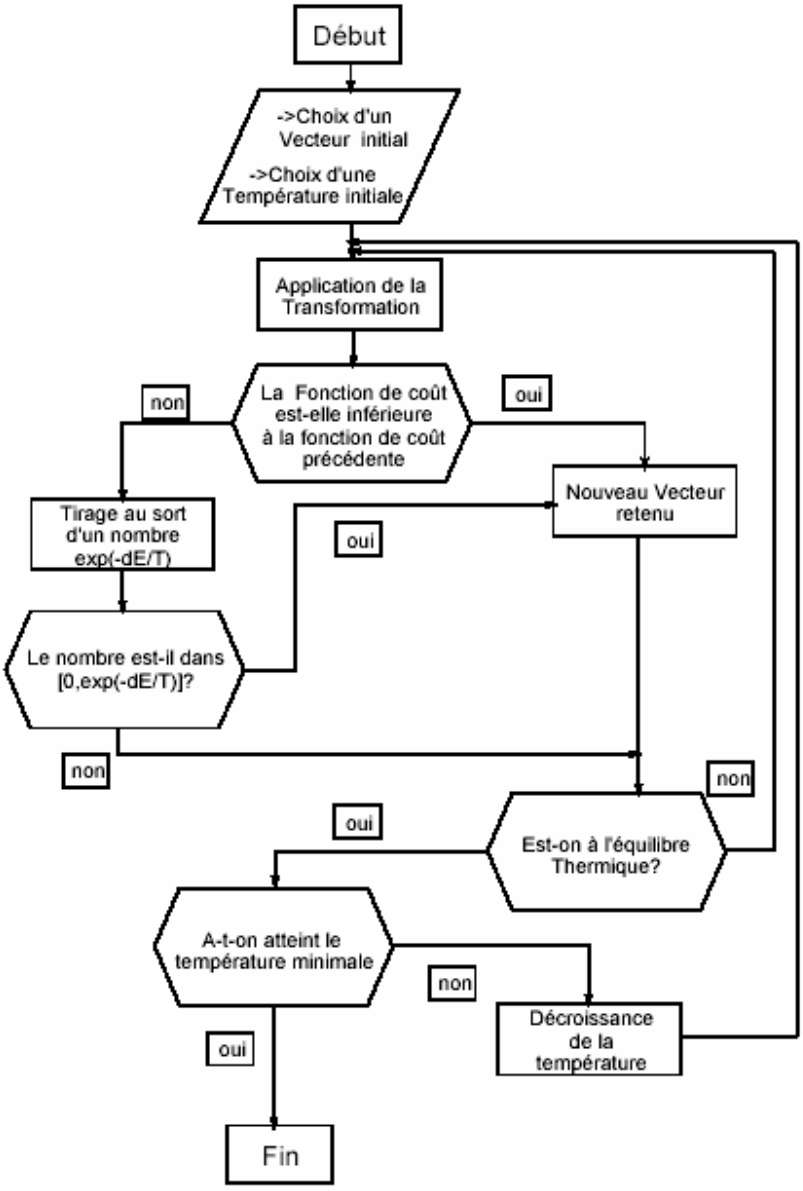


FIG 2 Algorithme du recuit simulé

e. Espace des configurations

Cet espace de recherche joue un rôle fondamental dans l'efficacité de l'algorithme. Cet espace est doté d'une topologie ou paysage d'énergie. Tout le problème réside dans le parcours de ces vallées et plateaux qui sont autant de minima locaux et de zones mortes.

Le choix de la fonction de coût ainsi que les déplacements élémentaires rentre dans une large part aussi dans l'efficacité de l'algorithme.

f. Avantages et inconvénient de la méthode

1. Les plus Elle procure une solution de bonne qualité. C'est une méthode générale applicable à un grand nombre de problème et facilement programmable. Elle offre une grande souplesse d'emploi et de nouvelles contraintes peuvent être facilement ajouter au programme.

2. Les moins Le nombre de paramètre, important à régler (température initiale, taux de décroissance, durée des paliers de température, critère d'arrêt...). L'algorithme garde un aspect empirique. Le temps de calcul peut être rédhibitoire.

5. Apprentissage par les Algorithmes Génétiques

Introduction

Il existe différentes sortes de méthodes d'apprentissage pour les réseaux de neurones : Deux classes de rétropropagation pour les RNR.

- Rétropropagation à travers le temps : dépliage temporel du réseau récurrent et application de l'algorithme.
- Rétropropagation récursive ou dynamique (temps réel)

Le Recuit simulé occasionne des temps de calcul prohibitif
Nouvelle approche : application des théories de l'évolution sur les RNR.

Les algorithmes génétiques sont appliqués à la recherche d'architectures adaptées aux problèmes posés ainsi qu'à la recherche des paramètres.
Pour les réseaux récurrents il y a deux facettes pour l'apprentissage

- Détermination de l'architecture par les Algorithmes génétiques
- Détermination des poids du réseau par différents algorithmes stochastiques (recuit simulé, Sollis et Wets).

Les Algorithmes Génétiques sont utilisés pour le développement des connexions récurrentes. Les algorithmes stochastiques peuvent être utilisés du fait qu'ils ne nécessitent pas de fonctions de transfert particulières. De plus ils facilitent une implémentation en ligne.

Comme dans tout type de réseau, l'apprentissage se résume à la minimisation d'une fonction de coût afin de trouver un minimum global. Ces méthodes garantissent l'obtention d'un optimal mais il reste le problème de la vitesse de convergence.

L'intérêt des GA est leur flexibilité, l'implémentation assez facile et modifiable à un large champ de problème. Mais il faut revoir l'implémentation à chaque problème spécifique. Il reste toujours le problème de réglage des coefficients de mutation, de sélection crossover...

5.1 GNARL Generalized Acquisition of Recurrent Links

Introduction

Comme défini précédemment les clefs d'un bon apprentissage avec les RNR est l'apprentissage de l'architecture (sa détermination) et l'adaptation des paramètres. Les méthodes courantes peuvent être classées en deux types:

- Les algorithmes constructifs : on démarre une structure simple et on ajoute des neurones nécessaires.
- Les algorithmes destructifs : on démarre d'un grand réseau et on taille les parties inutiles.

Le problème avec ces méthodes, elles ne permettent pas d'explorer toutes les configurations possibles de structures.

D'où le développement du **GNARL**. Cet algorithme permet d'acquérir en même temps la structure et les paramètres. Utilisation de la programmation évolutionniste.

5.2 Développement des réseaux connexionnistes

La programmation évolutionniste permet d'avoir une collection d'algorithmes pour l'apprentissage structurel et paramétrique des R.N. Ces algorithmes se distinguent parce qu'ils sont basés sur une population de position de recherche dans l'espace et non pas d'une seule position. Pendant un cycle de recherche tous les membres de la population sont classés par rapport à une fonction de coût. Les meilleurs sont sélectionnés pour être les parents de la prochaine génération. La progéniture est créée à partir de reproductions heuristiques spécialisées des parents.

Les algorithmes génétiques sont la forme la plus connue des théories évolutionnistes. Ils sont basés principalement sur le croisement.

5.3 Développement des réseaux de neurones par les GA

Les GA créent de nouveaux individus en recombinant les composants de deux membres de la population. Les GA se basent sur deux espaces de représentation distincts

- Espace de recombinaison : typiquement chaîne de bits, la structure sur laquelle les opérateurs génétiques sont appliqués.
- Espace d'évaluation : groupe de structures qui sont capables de faire la tâche.

Entre ces deux espaces il y a une fonction d'interprétation. Cette représentation duale est intéressante car on ne peut pas faire la recherche dans l'espace d'évaluation directement. Si on suppose que la fonction d'interprétation existe, on fait la recherche dans l'espace de recombinaison. On fait alors la deuxième supposition : le fait d'avoir la meilleure recombinaison au sens de la fonction d'interprétation entraîne le meilleur individu dans l'espace d'évaluation. Mais la fonction d'évaluation n'existe pas forcément.

5.4 Problème du croisement (crossover)

Développement de réseaux qui partagent la même topologie et les même poids : le croisement va créer une progéniture qui aura des composants redondants et perdra les capacités de leurs parents → dégradation.

Développement de deux réseaux qui ont une topologie identique mais des poids différents : Il est reconnu que pour une seule topologie plusieurs solutions sont possibles en termes de poids pour une même tâche. Mais chacune est implémentée par une représentation distribuée dans les couches du réseau.

Quand les parents diffèrent trop topologiquement : lors de la recombinaison des composants, la progéniture crée peut avoir une représentation trop distincte de l'un et de l'autre des parents (entre les deux) et peut donc se révéler incompatible

5.5 Réseaux et programmation évolutionniste

A la différence des A.G., la programmation évolutionniste définit des opérateurs de mutations, qui créent la progéniture avec une comparaison avec les parents. L'opérateur de mutation modifie les paramètres du réseau η par l'équation $\omega = \omega + N(\alpha, \varepsilon(\eta)) \forall \omega \in \eta \, ou \, \omega$ est un poids; $\varepsilon(\eta)$ erreur du réseau sur la tâche; α est une constante; $N(\mu,\tau^2)$ est une variable Gaussienne.

Contrairement aux A.G. cette théorie manipule directement le réseau et n'est pas obligée de passer par la représentation duale (recombination, évaluation). En évitant le crossover entre les réseaux on conserve l'individualité de la progéniture.

5.6 L'algorithme GNARL

C'est un algorithme basé sur les théories de l'évolution qui construit des Réseaux de Neurones en tenant compte de l'architecture et des paramètres de son apprentissage. La topologie de ces réseaux est des plus générales sans se limiter aux architectures uniformes et symétriques. Ici l'architecture reflète beaucoup plus la tâche à effectuer.

Sélection, reproduction, et mutation des réseaux.

Le GNARL initialise une population de R.N. générée aléatoirement (le nombre de neurones cachés est choisi dans une distribution uniforme définie par l'utilisateur; idem pour le nombre de connexions initiales).

Les neurones touchés par une connexion sont choisis en concordance avec les mutations de structure. Quand une topologie est choisie, tous les poids sont initialisés de manière aléatoire [-1; +1]. A chaque génération de recherche, les réseaux sont évalués par une fonction de coût définie par l'utilisateur. Les réseaux qui sont les meilleurs (1^{re} moitié en prenant les erreurs classées) seront les parents de la prochaine génération. Les autres sont détruits.

La méthode de sélection est utilisée dans beaucoup d'algorithmes de calcul évolutionniste bien que d'autres méthodes plus compétitives aient vues le jour. La génération de la progéniture se décompose en trois étapes :

- Copiage des parents
- détermination de la mutation à effectuer
- mutation des copies

Les mutations de réseaux sont séparées en deux classes qui correspondent aux types d'apprentissage :

- mutation paramétrique (change les valeurs des poids)
- mutation structurelle (change le nombre de neurones et les connexions; change directement l'espace des paramètres)

Sévérité de mutation

Le taux de mutation d'un parent η est dicté par la température du réseau. $T(\eta) = 1 - \frac{f(\eta)}{f_{\max}} où\ f_{\max}$ est le coût maximum pour une tâche. La température est déterminée en fonction de la concordance du réseau à être solution de la tâche. Cette mesure de performance du réseau est utilisée pour recuire les similitudes paramétriques et structurelles entre les parents et la progéniture. Un réseau avec une grande température sera beaucoup muté alors que ceux avec une température plus basse le seront moins. On part d'une recherche grossière qui s'affine petit à petit.

5.7 Mutation paramétrique des réseaux

Cette mutation est effectuée en perturbant chaque poids ω du réseau η avec un bruit Gaussien. Les poids sont modifiés comme suit : $\omega = \omega + N(0, \alpha T(\eta)) \forall \omega \in \eta \, où \, \alpha$! est une constante proportionnelle et *N(0; αT(η))* une loi normale gaussienne.

On doit occasionnellement avoir de larges mutations paramétriques afin d'éviter les minima locaux pendant la recherche. Ceci affecte la capacité de la progéniture. Afin de compenser cette perte d'efficacité, la mise à jour des poids est faite avec une variante de l'équation : $\hat{T}(\eta) = U(0,1)T(\eta) \, où \, U(0,1)$ est une variable aléatoire uniforme sur [0; 1]. Cette modification diminue la fréquence des larges mutations paramétriques sans les exclure complètement.

5.7.1. Mutation structurelle du réseau

Cette mutation change le nombre d'unités cachées du réseau et les connexions entre les neurones. Afin d'éviter les sauts critiques dans la minimisation de la fonction de coût, les mutations structurelles tendent à préserver le comportement du réseau. Les nouvelles connexions sont initialisées avec des poids nuls, conservant le comportement du réseau. Dans le même ordre d'idée, les neurones ajoutés dans la structure sont sans incidence sur les connexions. Elles seront ajoutées à la suite des mutations structurelles suivantes qui détermineront comment insérer la cellule ajoutée.

Mais cette notion de continuité est assez difficile à mettre en place. Exemple : enlever un neurone revient à modifier complètement la structure du réseau autour de ce neurone (liens + poids). La suppression d'un lien supprime le poids qui lui est attaché. De plus il faut s'intéresser à la classe du neurones et de ses liens (neurones d'entrées, cachés, de sortie). Quand il y a trop de différence entre le nombre de neurones cachés et le nombre de neurones d'entrée et de sortie, le processus de sélection doit être biaisé. Afin d'augmenter l'efficacité de la mutation structurelle on peut encore modifié l'équation d'adaptation comme suit $\Delta_{\min} + [U(0,1)\hat{T}(\eta)(\Delta_{\max} - \Delta_{\min})]$ $[\Delta_{\max}, \Delta_{\min}]$ un intervalle donné pour une mutation.

5.7.2. Modification du réseau pour adaptation

Afin de déterminer un réseau pour une tâche, l'algorithme GNARL n'a pas besoin d'un vecteur cible explicite, seule l'information de la fonction de coût est nécessaire. Mais si un tel vecteur existe on peut modifier la mesure de l'adéquation du réseau par les calculs suivants (de la moins pénalisante à la plus pénalisante.)

$$\sum_i (y_i - Sortie\ (\eta, x_i))^2$$

$$\sum_i |y_i - Sortie(\eta, x_i)|$$

$$\sum_i e^{y_i - Sortie(\eta, x_i)}$$

Le choix de la fonction d'adéquation n'altère en rien l'algorithme GNARL. Utilisation de cet algorithme pour la recherche de nourriture par les fourmis

5.8. Codage génétique des réseaux de neurones

5.8.1 Codage direct

a) Une représentation génétique: adaptée peut faciliter la définition des opérateurs génétiques. Fullmer grandement et Miikkulainen [Fullmer et Miikkulainen, 1991] ont utilisé un codage proche du génome C'est une suite d'entiers, semblables aux codons biologiques qui réprésentent les paramètres d'un noeud du réseau: naturels, un marqueur de début, une clé d'identification, la valeur de sortie initiale du réseau et une liste connexions de type clé de la source- poids de la connexion et enfin un marqueur de fin. Cette représentation facilite la définition des opérateurs qui peuvent être identiques à ceux d'un algorithme génétique, simple: les mutations changent un codon, le croisement génétique échange des sous chaînes de codons entre chromosomes.

b) Le croisement: dans des algorithmes permettent l'échange de blocs entre chromosomes évolutionnistes. L'idéal serait de pouvoir échanger des blocs fonctionnels de façon à ce que le résultat du croisement soit susceptible de caractéristiques différentes issues de chacun de ses profiter ce qui est loin d'être évident avec un codage direct. parents, La simple définition d'un opérateur de croisement n'est pas non plus un problème simple, à tel point que dans de nombreux cas, le croisement n'est même pas utilisé [Pasemann et Dieckmann, 1997], [Angeline et al. ,1994], [Yao et Liu, 1997]. et Poli [Pujol et Poli, 1997]utilisent une représentation sous forme de grille Pujol du réseau. Deux noeuds sont choisis et les liens, représentés sous forme relative (connexions au neurone de la ligne (L-1), 2 colonnes après (C+2), par exemple), sont précédents d'un des noeuds à l'autre. Les auteurs n'ont fait évoluer trans-posés que des réseaux de type perceptron, mais cette approche doit pouvoir se transposer à des structures quelconques. La limitation réside cependant dans l'obligation d'utiliser principale une grille de taille fixe, imposée initialement. *NEAT* [Stanley et Miikkulainen ,2001] apporte une solution complètement différente et originale à ce Chaque connexion dispose d'un numéro d'innovation attribué de façon unique lors de l'apparition de la connexion. Ce numéro d'innovation, permet de définir un opérateur de : sont similaires les chromosomes disposant d'un certain similarité nombre de connexions ayant le même numéro d'innovation. Cette similarité permet de faire la correspondance entre des connexions et d'échanger l'information génétique de

historiquement issue d'une même mutation, connexions supposées exercer une fonction comparable dans les différents réseaux. La définition d'une similarité permet également de définir des niches écologiques permettant de protèger les structures en utilisant le partage de fitness explicite nouvelles *fitness sharing* en anglais) [Goldberg et Richardson, 1987].

5.8.2. Codage indirect

Dans les méthodes décrites la correspondance entre un chromosome et précédemment, le réseau de neurones est directe et la traduction immédiate. Pour résoudre les problèmes cités ci- dessus ainsi Cependant, que d'autres problèmes, comme le passage à l'échelle, il est à considérer des codages pour lesquels la correspondance intéressant génotype-phénotype n'est plus immédiate, mais nécessite une phase de développement. Gruau a prouvé l'efficacité d'approches indirectes par à des approches basées sur un codage direct [Gruau et al, 1996], rapport de récents travaux [Stanley et Miikkulainen ,2001]suggèrent qu'un codage direct cependant peut être plus efficace qu'un codage indirect, montrant ainsi qu'au moins pour des réseaux de taille réduite, la différence entre codage direct et indirect n'est pas aussi nette que Gruau l'a suggéré.

a) Règles de réécriture: Un graphe peut être obtenu à partir d'une cellule initiale par exécution d'un programme ou de règles de production. Lindenmayer a développé une classe de fractales appelées L-Systems [Lindenmayer, 1968] pour modéliser le développement des plantes. Un L-Système est un système de réécriture de chaînes en parallèle. Il est composé d'un ensemble de règles de production qui définissent des transformations de chaînes élémentaires en d'autres chaînes élémentaires. Kitano [Kitano,1990] et Boers [Boers et al. ,1993] ont ce principe pour faire évoluer des réseaux de neurones.

b) Programmation génétique: La programmation [Koza, 1992] a inspiré de nombreuses méthodes de développement génétique de réseaux de neurones dont la plupart sont inspirées du *Codage Cellulaire* de Gruau [Gruau,1992], [Whitley et al. , 1995], [Gruau, 1994].
Le codage cellulaire part d'un réseau simple contenant des cellules de développement reliées aux entrées et sorties. Ces cellules exécutent leur programme de développement .Lorsque l'exécution atteint un symbole terminal, la cellule se transforme en neurone. Chaque cellule de développement d'une copie du programme de développement ainsi dispose que d'une tête de lecture qui pointe sur la prochaine instruction à exécuter. Les instructions de développement positionnent les différents paramètres du futur neurone et complexifient la part des instructions de division cellulaire. Ces instructions structures de division sont le point clé de cette méthode. Il en existe types. Elles répartissent les connexions entre cellule différentes mère et cellules filles de différentes manières: elles peuvent les recopier dans la nouvelle cellule, transmettre les connexions sortantes et créer

un lien entre la cellule mère et la cellule fille.

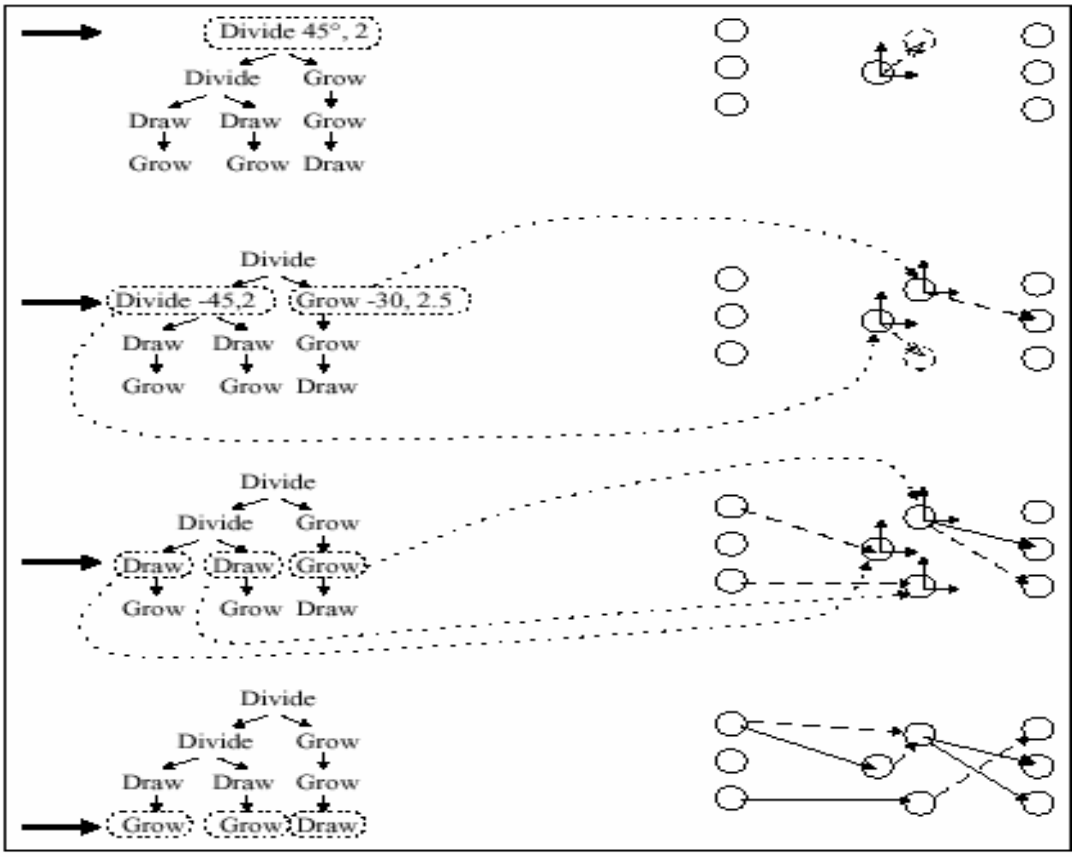


FIG 3 :**Exemple de programme SGOCE**

SGOCE [Kodjabachian et Meyer, 1995], [Kodjabachian et Meyer, 1997], [Kodjabachian, 1998] est une méthode inspirée du codage cellulaire dans laquelle les cellules de développement sont situées dans un substrat à 2, voire 3, dimensions. Cela permet de créer des connexions depuis ou vers des neurones par des opérations autres que des divisions: il est possible de viser une zone du substrat et d'attirer ou de créer une connexion depuis ou vers la cellule la plus proche de cette zone (figure 3).
Gruau utilise un programme génétique L*'Edge Encoding* [Luke et Spector, 1996] est une autre variante du codage dans laquelle le développement n'est plus basé sur cellulaire des cellules de développement, mais sur des liens, des connexions entre cellules. Ce sont ces liens qui exécutent le programme de développement.

c) Neurogénèse: Si les méthodes évoquées restent relativement éloignées de la réalité biologique, précédemment essaient d'y être beaucoup plus fidèles et mettent en place certaines un développement dépendant d'émissions de médiateurs L'évolution a alors pour tâche de régler ces émissions chimiques, et de choisir le comportement adapté en fonction des locales de différents agents chimiques, voir par concentrations exemple [Astor,1998], [Michel et al. ,1997] et [Harvey,1993].

6. Conclusion

Les algorithmes presentés dans ce chapitre contrairement aux algorithmes de rétro-propagation peuvent utiliser des fonctions d'activation non dérivables. Ils permettent aussi une recherche non monotone de structure de réseaux de neurones. Et une grande nouveauté permet la manipulation directe de la structure du réseau, contrairement aux autres algorithmes qui partent d'une structure donnée et qui ne font que l'optimisation des poids sur cette structure.

Ce chapitre va nous servir à comprendre mieux les techniques de modélisation, le contrôle et l'évolution des entités du problème de poursuite. Il faut noter que tous les contrôleurs neuronaux utilisés dans la robotique, la poursuite et la vie artificielle sont des réseaux dynamiques récurrents.

Identification et commande de système par réseau de neurones récurrents

1. Introduction

La diversité des systèmes non linéaires empêche une résolution simple pour leurs problèmes de commandes, alors que même des systèmes linéaires posent des problèmes d'optimisation, donc on voit la nécessité d'utiliser des techniques d'optimisation comme le recuit simulé ou même les réseaux de neurones.

La capacité des réseaux de neurones à modéliser des systèmes non linéaires est largement exploitée pour la synthèse de contrôleur [Cybenko, 1988], [Lapedes et Farber, 1987]. Mais des problèmes subsistent :

- Pas de voie systématique pour déterminer la structure pour modéliser un système donné
- La rétropropagation n'assure pas la convergence de l'apprentissage
- Le réseau de neurones est une boite noire

De plus c'est le Perceptron multicouche ou MLP qui est le plus souvent utilisé. Ce modèle de réseau de neurones n'est pas dynamique. On le rend dynamique en lui administrant en entrée, ses entrées et ses sorties aux instants précédents. On peut alors l'interpréter comme un modèle NARMAX (**N**on-linear **A**uto-**R**egressive **M**oving **A**verage with e**X**ogenous inputs). Mais on réduit alors de beaucoup les dynamiques contenues dans ce type de réseau par rapport à un réseau récurrent.

2. Approche de commandes neuronales

La plupart des méthodes de commandes neuronales sont basées sur la commande inverse, bien connue dans les disciplines de l'électronique, automatique et robotique.

2.1. Technique de commandes neuronales basées sur le système inverse

Commande neuronale basée sur le copiage : l'imitation ou le copiage est une caractéristique des systèmes biologiques. La première méthode de commande neuronale consiste à dupliquer le contrôleur humain. Cette technique a été appliquée au pendule inversé en 1964. Cette méthode se révèle utile pour le contrôle de processus industriel.

L'apprentissage consiste en un apprentissage de la correspondance entre les informations reçues par le contrôleur humain et la commande d'entrée. La principale difficulté de cette méthode est de déterminer les informations utilisées par le contrôleur humain (Cf. fig. 1 et 2).

2.2. Apprentissage de commande neuronale directe inverse

Cette commande inverse a pour but la commande du système en utilisant ses dynamiques inverses. Le réseau reçoit en entrée les sorties du système (Cf. figure 1.24) L'apprentissage est effectué en donnant au contrôleur une séquence d'entrée u. Bien qu'il y ait un retour des sorties du système sur l'entrée du contrôleur, le système est globalement en boucle ouverte car on ne tient pas compte de la sortie du système et la sortie désirée. C'est à dire que le réseau de neurone ne fait rien pour rattraper la déviation entre la sortie réelle et la sortie désirée.

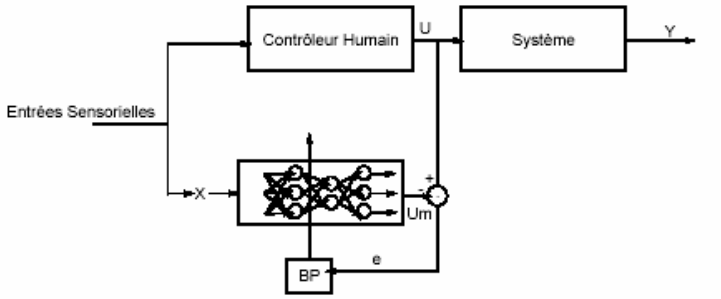


Fig. 1 : Copiage de l'expert humain par le neuro-contrôleur sans boucle de retour

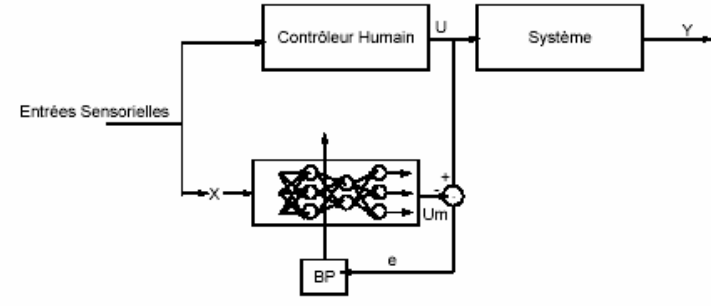


Fig. 2 : Copiage de l'expert humain par le neuro-contrôleur avec boucle de retour

Le plus dur reste donc à déterminer la séquence des entrées u du réseau de neurones. Le système doit être porté dans son espace de travail ou le contrôleur devra

effectuer la correction. Ceci est très difficile à assurer sans une connaissance a priori très importante du système.

2.3. Apprentissage de la commande neuronale inverse spécialisée

Ils proposent l'apprentissage spécialisé inverse. C'est une approche de commande neuronale dirigée par un but. C'est la différence fondamentale avec l'approche décrite précédemment. Le réseau opère un apprentissage en ligne afin de minimiser un critère de commande (idem le critère d'apprentissage) $e_y = r-y$ (Cf. fig. 3)
L'erreur e_y n'est pas directement relation avec la sortie du contrôleur u comme cela

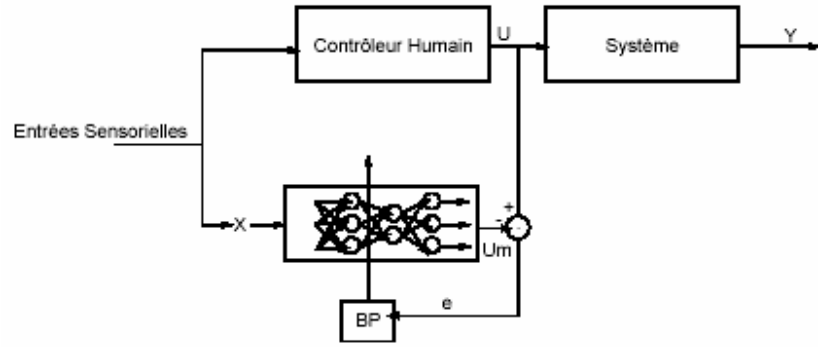


Fig. 3 : Architecture de commande neuronale directe

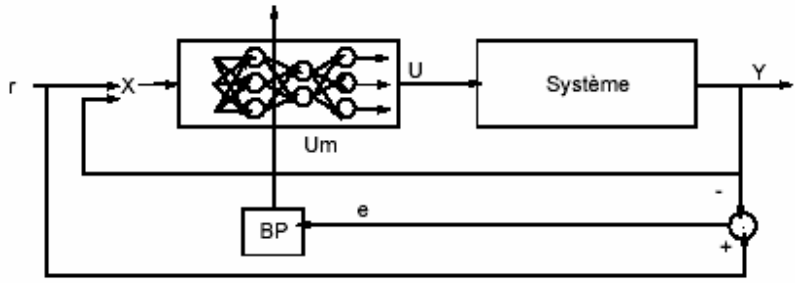


Fig. 4 : Architecture de commande inverse spécialisée

devrait être. De ce fait cette approche ne peut mener vers une solution optimale. D'ici ey peut être totalement non corrélée au critère de performance d'apprentissage de la commande neuronale $e_u = u - \bar{u}$.

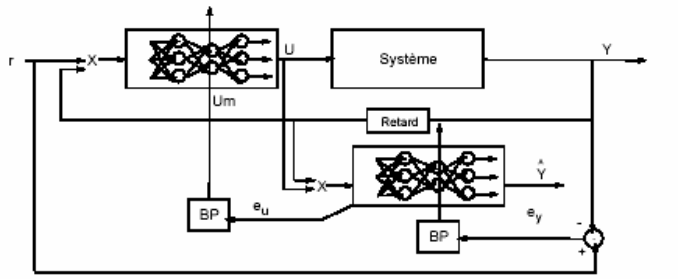


Fig. 5 : Architecture de commande neuronale avec rétropropagation à travers le temps

devrait être. De ce fait cette approche ne peut mener vers une solution optimale. D'ici ey peut être totalement non corrélée à au critère de performance d'apprentissage de la commande neuronale eu =u-$\bar{u}$.

2.4. Apprentissage de contrôleur neuronal par rétropropagation à travers le temps

Pour éviter le problème de la commande spécialisée inverse, des chercheurs (Narendra, Jordan, Rumelhart, Nguyen) ont développé indépendamment la rétropropagation à travers le temps. Le problème de l'apprentissage de la commande spécialisée inverse est que l'erreur de performance ey = r-y n'est pas fiable. En effet elle n'est pas apparentée directement à la sortie u de neuro-contrôleur. Un erreur faible d'apprentissage du neuro-contrôleur serait donnée par eu =u-$\bar{u}$. Les chercheurs ont donc proposé d'émuler eu en utilisant la rétropropagation de ey à travers le modèle du système. Dans ce sens l'erreur ey rétropropagée vers la couche d'entrée du modèle feed-forward devrait correspondre à l'erreur eu =u-$\bar{u}$ (en supposant que l'entrée u a un effet linéaire sur le système).

L'apprentissage de commande peut donc être réalisé dans différents cas.

On peut remarquer que le modèle neuronal du système est entraîné avant le contrôleur (Cf. fig. 5). Bien que l'erreur soit rétropropagée dans le modèle neuronal ce dernier n'est pas adapté à l'apprentissage du contrôleur. Il peut être utile dans certaines utilisations de choisir la sortie du modèle neuronal ŷ plutôt que la sortie réelle (lorsque le système ne peut être utilisé lors de la phase d'apprentissage du correcteur).

L'approche étudiée par Narendra est un peu différente car il injecte un modèle de référence dans la structure pour donner une sortie transitoire voulue $\bar{y}$ au système plutôt qu'une sortie désirée r qui ne dépend pas du temps. C'est alors cette sortie voulue que l'on injecte dans le neuro-contrôleur (Cf. fig. 6).

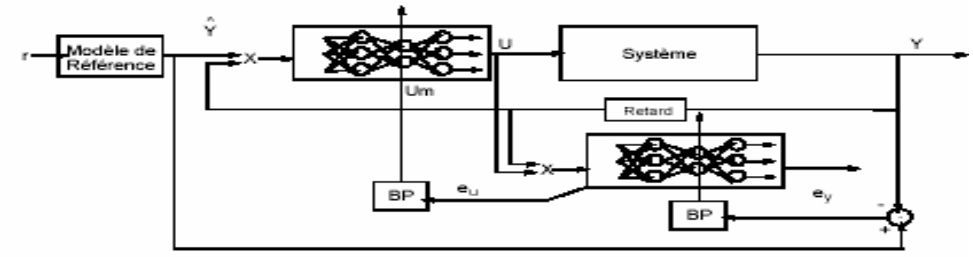


Fig. 6 : Architecture de commande neuronale avec rétropropagation à travers le temps et modéle de référence

2.5. Commande biologique et commande neuronale basées sur le système inverse

Les méthodes de commande neuronale basées sur le système inverse qui sont appliquées le plus souvent en robotique ont de sérieux inconvénients. Le plus important concerne le manque de retour d'informations. Si le neuro-contrôleur n'est pas l'inverse exact du système alors il se peut que la commande soit sans effet ce qui ne surviendrait pas s'il y avait un retour d'information.

Mais il n'est pas simple d'apprendre l'inverse des dynamiques très retardées d'un système bruité. De plus même si le modèle neuronal est assez précis on se rend compte qu'il est très sensible au bruit.

Le second problème vient du fait que tous les systèmes ne possèdent pas d'inverse. Ces problèmes restreignent donc les domaines d'application de cette méthode. Le retour d'information (feedback ou boucle de retour) possède des propriétés très importantes et doit faire partie d'un processus de commande. Mais alors le problème se complique car a boucle de retour peut entraîner des instabilités.

Dans les systèmes humains on trouve différents contrôleurs. Il semblerait que les systèmes en boucle ouverte et boucle fermée jouent des rôles différents dans la commande humaine. Pour les commandes d'actions rapides les systèmes en boucles ouvertes sont mieux adaptés. Mais pour les actions précises se sont les systèmes bouclés qui sont le mieux adaptés.

Une remarque importante concernant les systèmes humains est qu'avec de l'apprentissage les systèmes de commandes biologiques développent un modèle interne précis du système à commander. Les systèmes humains ont une grande capacité de généralisation par rapport à des situations inconnues.

3. Commande prédictive basée sur le modèle neuronal

La commande prédictive a été développée principalement pour surmonter le problème des retards dans les systèmes complexes qui ne réagissent pas immédiatement à

l'application d'une commande. Le retard du système est le temps que le système met pour réagir.

Le contrôleur ne change pas d'état pendant ce retard. Ceci peut induire des effets indésirables. Une solution consiste donc à attendre le temps du retard et seulement après de lancer le correcteur. La deuxième solution est la meilleure consiste à utiliser un modèle prédictif du système qui prédira la sortie du système à la suite du retard. A ce moment on peut donc anticiper la commande. Ceci est donc la base de la commande prédictive. On détermine la commande en fonction de la sortie estimée du système par le modèle. Cette idée a été généralisée par Clark en 1987 et appliquée aux systèmes à dynamiques complexes et appelé le "Generalized Predictive Control" GPC. C'est la méthode principale pour la détermination de correcteur basé sur la commande prédictive (Cf.fig 7).

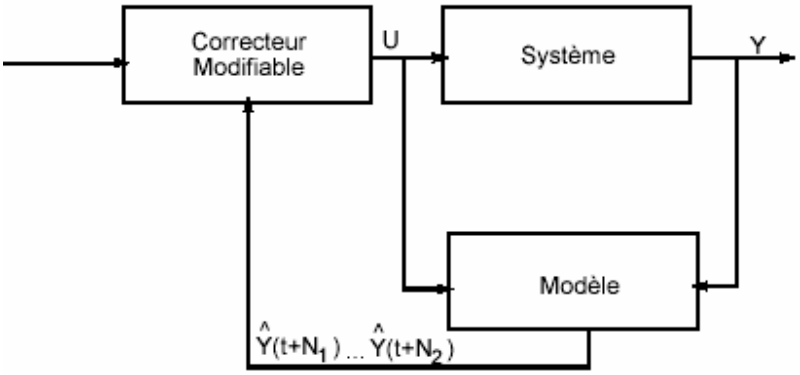


Fig. 7 : Structure générale d'un correcteur basé sur un modèle prédictif

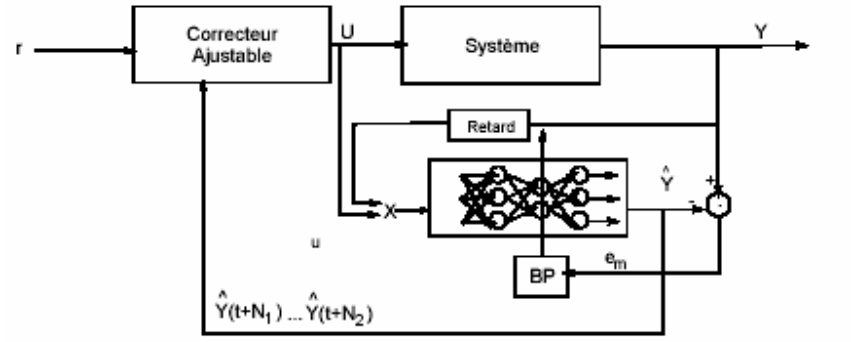


Fig. 8 : Structure générale d'un correcteur basé sur un modèle prédictif neuronal

("Model Based Predictive Control" MBPC). Ce type de correcteur est composé de trois parties :

- Le système
- Le modèle
- Une fonction d'optimisation

Le modèle prédit les sorties du système. Par rapport à ces sorties, la fonction d'optimisation détermine la séquence de commandes la plus adaptée au système. La fonction d'optimisation prend la forme d'une fonction de coût quadratique :

$$J(N_1, N_2, N_u, t) = \sum_{i=t+N_1}^{t+N_2} \mu(i)\,[\bar{y}(i) - \hat{y}(i)]^2 + \sum_{i=t}^{t+N_u} \lambda(i)\,[\Delta u(i-1)]^2$$

où $\bar{y}$ est la sortie désirée, ŷ est la sortie du modèle, Δu(i) = u(i) - u(i-1) est le pas d'entrée du procédé, N1 et N2 définissent l'horizon de prédiction, N_u est l'horizon de commande et λ et μ sont des constantes qui pondèrent le comportement futur.

La caractéristique principale de cette méthode est que la conception d'un correcteur n'est pas nécessaire. Les réseaux de neurones MBPC ont été largement utilisés dans la modélisation de procédés. Ceci donne un correcteur basé sur un modèle neuronal prédictif (Cf. fig. 8). Les réseaux de neurones peuvent aussi être utilisé comme correcteurs.

Remarque sur les techniques de commandes neuronales

La plupart du temps les MLP perceptrons multicouches sont utilisés avec tout ce qu'ils comportent comme défaut. On ne peut déterminer avec précision les propriétés de commandes et de modélisation à cause de leur concept de boite noire. Ce qui est très contraignant car on doit souvent faire référence aux propriétés de contrôlabilité et commandabilité pour établir sa stabilité et sa robustesse.

De plus les méthodes d'apprentissage comme le gradient sont lentes et leurs convergences ne sont pas prouvées. Ceci est du aux minima locaux et aux zones mortes.

4. Commande par réseaux de neurones modulaires sélectionnés

Dans les réseaux modulaires qui sont facilement modifiables, on peut réutiliser les modules pour différentes tâches au lieu d'apprendre les parties communes de toutes les tâches. La difficulté principale dans la conception de réseaux de neurones modulaires est la décomposition de la tâche globale en sous tâches facilement codables sous forme de réseaux de neurones. La modélisation en sous tâches doit donc être un processus interactif entre les capacités de calcul de chaque module et la complexité des sous tâches.

Peu d'algorithmes peuvent faire la décomposition en sous tâches de manière systématique. Ces algorithmes se caractérisent donc par des systèmes de portes dans l'architecture. A chaque instant le système de sélection de modèle opère une sélection des modules les plus aptes pour le calcul. Ces réseaux de neurones modulaires sont appelés

les "Gated Modular Neural Networks GMNN". On trouve dans la littérature plusieurs types de réseaux de neurones que l'on peut distinguer par le critère utilisé pour la décomposition de l'espace d'entrée en sous-espace.

Les système de sélection de modèle par groupage spatial les GMNN définis ici opèrent un groupage spatial des entrées de façon à sélectionner et répartir les vecteurs d'entrées en différents modules. L'algorithme le plus abouti en ce sens est le "LMN" "Local Model Network" , lequel a été étendu par Johansen et Foss en 1992 pour la modélisation et la commande. Le LMN est en fait une généralisation des réseaux neuronaux à fonctions de bases développer au départ pour des tâches de classification. De plus il possède des ressemblances frappante avec le modèle flou de [Takagi et Sugeno, 1985].

L'architecture d'un réseau à modèle local Ce réseau consiste en un réseau mono couche et un système de (porte) sélection de chemin souvent composés par des fonctions radiales de bases (RBF). Les différents modèles locaux sont activés par un vecteur d'entrée X qui varie dans le temps, ce qui correspond souvent aux vecteurs des paramètres d'un modèle NARMAX du système. Chaque modèle local peut être considéré comme un modèle NARMAX local du système (Cf. fig. 9).

La sortie de chaque modèle sélectionné est pondérée par l'activité w des RBF connectée. De là, la sortie du réseau est la somme des sorties pondérées de tous les modèles.

A chaque instant le système de sélection de modèle opère un groupage hyper sphérique de l'espace d'entrée afin de donner le vecteur d'entrée Φ aux meilleurs modèles. Le vecteur Φ est le plus souvent un sous-ensemble du vecteur d'entrée X. Le groupage est effectué par des fonctions radiales de bases (RBF) qui ont pour caractéristique un centre et une largeur. L'activation d'une RBF est faite par (Cf. fig. 10)

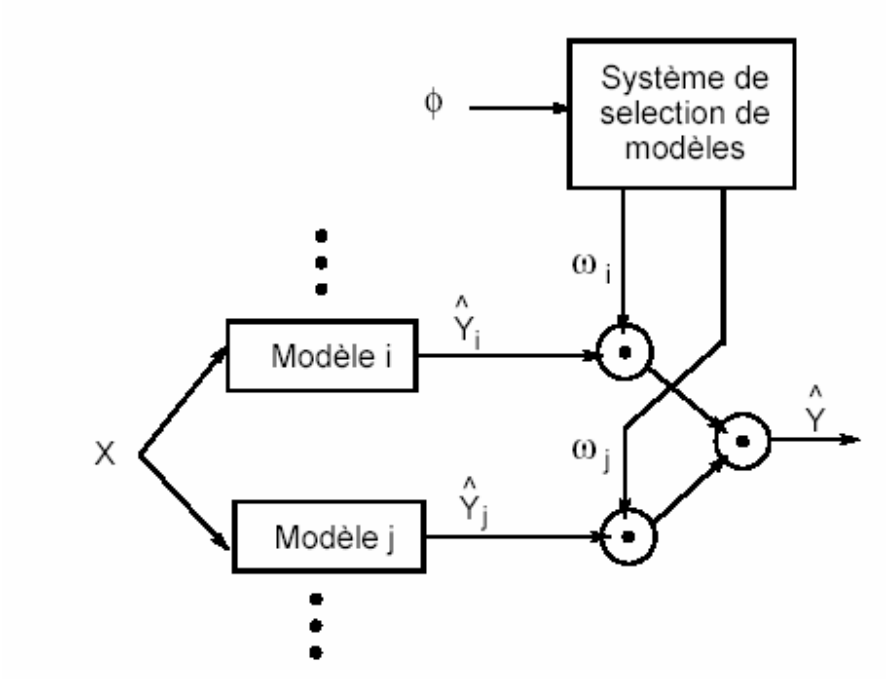


Fig. 9 : Le réseau à modèles locaux ŷ

Le calcul d'une distance Euclidienne entre les coordonnées spatiales du vecteur clustérisé $\Phi = [\Phi 1 \Phi 2]$ les coordonnées du centre Ci = [$c_{i1}c_{i2}$] de la fonction radiale de base RBF_i.

Un potentiel d'activation pi est déterminé en fonction de la largeur σ_i de la RBF_i $p_i = \frac{(\Phi - C)^2}{\sigma^2} où\, C_i = [c_{i1}c_{i2}] et [\Phi_1 \Phi_2]$ sont respectivement les coordonnées spatiales du centre de la RBF_i et l'entrée clustérisée.

Il est activé par un vecteur d'entrée ' qui est le plus souvent un sous-ensemble du vecteur d'entrée X qui active le réseau de modèles locaux. Le système de sélection de modèle local est composé de RBF utilisée pour le groupage de l'espace d'entrées (Cf. figure 1.32 et 1.33). Les sorties de chaque RBF pondèrent (ω) la sortie du modèle local connecté.

e^{-p_i} est utilisé pour obtenir un groupage non-linéaire qui a des caractéristiques Gaussiennes. L'activation de la fonction radiale de base i est $\omega_i = e^{-p_i} \cdot \omega_i$ sera proche de 1 si pi est très petit et proche de 0 si pi est très grand. L'idée est alors d'utiliser ω_i pour pondérer la sortie la sortie du modèle local ŷ= $f_i(X).\omega_i$ où $f_i(X)$ fait référence au modèle local i NARMAX.

La sortie du LMN est la somme des sorties des modèles locaux $y_{LMN} = \sum_{i=1}^{N} y_i . \omega_i$ où N est le nombre de modèles locaux. En résumé on peut dire que le comportement d'un LMN est la décomposition de l'espace du système réel en sous-espace plus petit déterminé par des RBFs. Des modèles locaux simples sont utilisés pour décrire le système dans chacune de ces régions. Un modèle global est formé par l'interpolation des modèles locaux en utilisant une fonction d'interpolation w qui dépend de la condition de fonctionnement Φ.

Le principal avantage est de pouvoir facilement transformer le réseau à modèle local en réseau à régulateur local.

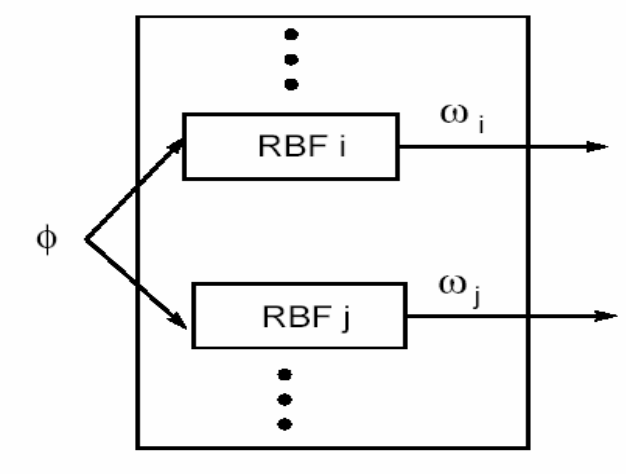


Fig. 10 : Le système de sélection de module

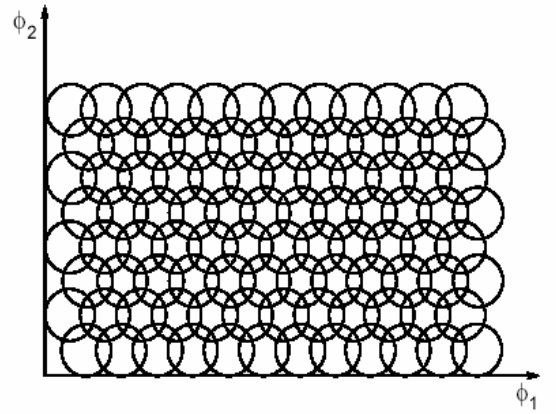


Fig. 11 : Exemple d'un groupage hypersphérique d'un espace d'entrée à deux dimensions

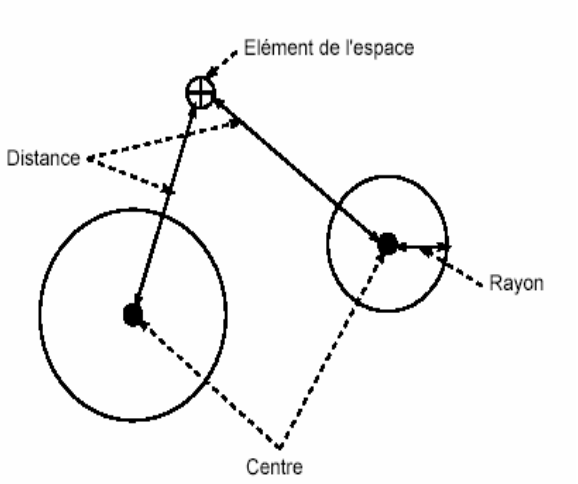


Fig. 12 : Description des composants de deux fonctions radiales de base

a. Adaptation des paramètres d'un réseau à modèle local

Il y a deux types de paramètres dans un réseau à modèle local. Les premiers sont les paramètres des réseaux locaux, les seconds sont les paramètres des RBFs connectées aux réseaux locaux. L'optimisation des paramètres des réseaux locaux est assez simple et on opte le plus souvent pour une méthode type des moindres carrés.

b. Apprentissage des modèles locaux

Il y a deux façons d'effectuer l'apprentissage des modèles locaux. La première est de considérer le réseau dans son ensemble et de faire l'estimation des paramètres des modèles locaux. La seconde entraîne de faire l'estimation des paramètres de chaque modèle local indépendamment des autres. C'est l'apprentissage local. Il y a plusieurs avantages à utiliser l'apprentissage local plutôt que le global :

- Murray et Smith 1994 ont montré que l'apprentissage local était plus performant en terme de modélisation. L'apprentissage local rend plus précis le modèle local sur sa région ce qui rend le modèle global encore plus performant.

- Un apprentissage local est plus simple à mettre en oeuvre. Les modèles locaux étant linéaires l'apprentissage peut se faire à l'aide d'une simple méthode des moindres carrés. C'est l'interaction entre les différents modules qui provoque les non-linéarités dans la structure globale. Dans le cas d'une optimisation globale et de l'utilisation d'une descente de gradient de l'erreur la convergence vers un minimum global n'est plus assurée et le temps d'apprentissage s'allonge.

5. Conclusion

Dans ce chapitre, on a vu quelques modèles de commande par des réseaux de neurone récurrents. Ces exemples sont très pratiques, on les trouve dans les commandes des robots, des avions, des missiles et des entités des jeux vidéo et de la réalité virtuelle. Ces derniers vont servir à la compréhension des techniques utilisées dans le chapitre de la poursuite de Cliff & Miller.

Le problème de la poursuite de Cliff&Miller ; concepts et extensions

1. Introduction

Dans le problème de poursuite évasion, un « poursuiveur » chasse un «poursuivi» dans un environnement plus ou moins complexe. En nature, les comportements de poursuite et d'évasion sont communs parce que les conflits de l'approche et de l'action d'éviter sont communs entre les animaux.

Ce domaine est ainsi considéré idéal pour étudier le comportement adaptatif à beaucoup de niveaux au-dessus de beaucoup d'échelles de temps [Miller et Cliff, 1994] en construisant le poursuiveur et les robots ou les programmes de poursuite. Beaucoup de tâches potentielles de robot sont considéré essentiellement comme des problèmes d'évasion de poursuite, par exemple. L'action d'éviter les obstacles et la navigation dirigée par but. Malgré que ce soit un domaine restreint, le problème de poursuite a des projections sur plusieurs autres secteurs, grâce à sa nature dynamique dans le temps.

Mathématiquement, le problème de poursuite est soit trivial dans le cas d'un mouvement newtonien avec des équations de mouvement connues à l'avance ou bien mathématiquement intraitable dans le cas où on a seulement une enveloppe de l'équation de mouvement des entités de la poursuite. En mathématique le problème de poursuite s'inscrit dans le domaine des jeux différentiels.

Cliff & Miller ont proposé un modèle pour la simulation du comportement de la poursuite entre un poursuiveur et un poursuivi. Ce dernier prend en charge tous les aspects qui composent la poursuite. En commençant la morphologie jusqu'au contrôle des entités générées. Ce modèle se base principalement sur le concept de la co-évolution des contrôleurs neuronaux des entités de la poursuite. Les contrôleurs neuronaux sont générés génétiquement par un processus de morphogénéisation, qui est très proche à l'évolution des plantes naturelles.

2. L'évolution versus la co-évolution

On va commencer par la définition des deux concepts, ainsi qu'une une discussion analytique, qui permet de comprendre mieux l'intervention de l'évolution et de la co-évolution dans le problème de la poursuite.

2.1 Définitions et concepts d'évolution :

1. L'évolution est un :
    - Processus de modification des êtres vivants au cours des générations. Ces modifications peuvent affecter le matériel génétique, les comportements ou la forme des individus. Ils peuvent résulter en l'apparition de nouvelles espèces.

2. L'évolution est un processus :

- Parallèle à la progression linéaire ininterrompue du temps qui, dans une alternance variable de phase de progressions stagnations ruptures régressions, achemine l'ensemble des facteurs du monde phénoménologique.
- Matière organique et inorganique, concepts et faits vers l'accomplissement de leur finalité, et qui oriente l'individu vers l'augmentation de ses connaissances, une meilleure compréhension de l'univers, de lui-même, de l'organisation de sa société et, en principe, vers un état général qui, dans des laps de temps variés, le propulsent vers sa propre conquête ainsi que celle de son environnement de plus en plus élargi.

2.2 Définitions et concepts de co-évolution :

La coévolution, c'est le processus sans fin dans lequel deux adversaires construisent sans cesse de nouvelles armes pour ne pas être distancé par « l'autre ».

Des exemples typiques de coévolution en biologie se trouvent dans les relations entre les agents pathogènes et leurs hôtes. La coévolution existe aussi dans les systèmes mutualistes, où chacun exploite l'autre. L'illustration la plus belle est celle des orchidées de Madagascar et de leurs papillons pollinisateurs. Les premières ont des tubes nectarifères de 30 cm de longueur et les seconds ont des trompes de 25 cm. L'allongement démesuré des nectaires et des trompes au cours de l'évolution s'explique seulement par un processus co-évolutif qui avait été déjà entrevu par Charles Darwin.

La question cruciale est celle du rôle de la coévolution dans le phénomène grandiose de l'évolution elle-même [Petter et De Jong, 1994]. N'est-elle qu'une anecdote, ou au contraire un mécanisme fondamental ? L'hypothèse dite de « la Reine Rouge » propose qu'elle soit la base même de l'aventure de la vie.

Avec l'homme apparaît une forme entièrement nouvelle de coévolution, non plus entre des objets vivants mais entre deux processus : le génome et la culture. Par

leurs traditions culturelles, transmises de génération en génération, les hommes influencent de plus en plus fortement les processus de la sélection naturelle. Par exemple, les progrès de la médecine contrarient certainement la sélection des gènes de résistance aux maladies. Quant aux interventions directes sur le génome, elles relègueront les processus naturels au rang d'accessoires obsolètes.

a. Définition de courses aux armements :

La lutte entre deux espèces exige que le développement de techniques et de méthodes pour mettre les deux espèces à l'abri de toute disparition possible, on appel ce développement courses aux armements. Le poursuiveur évolue pour attraper le poursuivi, alors que le poursuivi évolue en même temps pour échapper au poursuiveur [Miller, 1996]. Dans le commencement, ayez la basse exécution et le poursuiveur pourrait par exemple facilement évoluer un comportement qui attrapera la proie. Les espérances sont que les deux espèces augmentent graduellement leur comportement dans une complexité plus élevée ensemble qu'elles réaliseraient par eux des individus.

b. Définition de l'hypothèse de la Reine Rouge :

Leigh Van Valem qui a proposé il y a une trentaine d'années ce que l'on appelle l'hypothèse de la « Reine Rouge ». La « Reine Rouge » évoque notre enfance et « Alice au Pays des Merveilles » de Lewis Carrol. A un moment dans le conte, Alice et la Reine Rouge sont en train de courir et Alice s'aperçoit qu'autour d'elle le paysage ne change pas. Alors, elle interpelle la Reine Rouge et lui dit : « Comment se fait-il que le paysage ne change pas ? Et la Reine Rouge lui répond une phrase qui est devenue célèbre : « Nous courrons pour rester à la même place ». Eh bien, c'est le symbole de la course aux armements. C'est à dire que la course aux armements se produit, elle n'améliore pas l'adaptation mais chaque fois qu'une espèce s'adapte, elle oblige les autres espèces à s'adapter. Et en s'adaptant, il y a de nouvelles structures, donc de nouvelles complexités qui apparaissent. Et on peut penser que toutes les courses aux armements qui durent, il ne faut pas l'oublier depuis l'origine de la vie, c'est à dire depuis trois milliards et demi d'années, eh bien, elles ont abouti à la complexité.

c. Discussion :

En analysant les deux concepts en remarque que :

- L'évolution apparaît dans le processus de co-évolution de chaque espèce.
- La co-évolution ne peut se faire qu'entre deux ou plusieurs espèces alors que l'évolution est un processus pour une seule espèce.
- L'évolution peut être évaluée d'une génération à une autre par une fonction objective, mais par contre c'est très difficile de trouver une façon d'évaluer les résultats de la co-évolution des concurrents.
- On voit clairement que le problème de poursuite est très proche à la définition de la co-évolution qu'a l'évolution tout courte.

3. Les méthodes de simulation

Le simulateur de coévolution de la poursuite évasion a été développé et raffiné pour servir l'étude des courses aux armements co-évolutionnaires pour caractériser les stratégies comportementales résultantes et pour analyser les architectures fondamentales de sensori-moteur qui produisent des comportements observables par l'interaction avec l'environnement .La simulation rapproche un système spatio-temporel continu en mettant à jour un modèle d'animats dans leur environnement simulé. Ici nous récapitulons certaines des sections après de détails les plus significatives.

Les sections suivantes décrivent les équations régissant la dynamique physique du mouvement pour les neurones artificiels de l'animat, le codage génétique utilisé, et l'algorithme génétique avec tous ses opérateurs notamment un opérateur important qui est la duplication.

3.1 La modélisation physique des entités du problème

a. Définition de l'animat :

Les animats sont des agents simulés sur ordinateur ou des robots réels - dont les lois de fonctionnement sont plus ou mois étroitement inspirées de mécanismes naturels, et dont les comportements exhibent certaines des caractéristiques d'autonomie et d'adaptation dont les animaux sont capables. [Meyer et al, 1991]

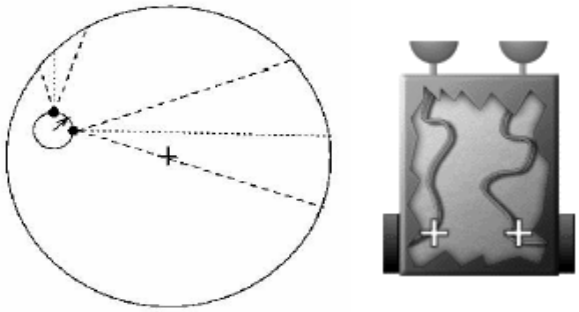


Schéma 1 Les animats

b. La dynamique physique

On utilise une simulation bidimensionnelle où les animats circulaires se chassent dans un plan infini vide sans obstacles ni frontière. En dépit du manque d'une troisième dimension spatiale. Nous avons utilisé un modèle physique assez réaliste.

Le contrôleur neuronal pour chaque animat donne deux valeurs produites désignées sous le nom du $\mathbf{v_l}$ et du $\mathbf{v_r}$: sont traités respectivement comme les entrées gauche et droit des signaux du moteur, (c'est à dire que le taux de changement d'angle d'orientation dépend de la différence entre les entrées gauche et droit de moteur, comme dans beaucoup de robots réels).l'équation de mouvement sont basés sur la physique newtonienne point .

Les équations appropriées pour le mouvement de translation des valeurs de sortie vers l'accélération est (***f*** représente la position *x,y* ***)***:

$$m\frac{d^2f}{dt^2} + c_f\frac{df}{dt} = k_f\frac{v_l + v_r}{2}$$

Pour le mouvement rectiligne l'avant, où **m** est la masse d'animat, $\mathbf{C_f}$ est un coefficient frottement et le $\mathbf{k_f}$ est une constante structurelle, et

$$\vartheta\frac{d^2\alpha}{dt^2} + c_\alpha\frac{d\alpha}{dt} = k_\alpha(v_l - v_r)$$

Pour changer l'angle d'orientation **α** où **υ** est le moment d'inertie d'animat du c_α est un coefficient angulaire de frottement, et le $\mathbf{k_\alpha}$ est une constante structurelle.
L'utilisation de ces équations réalistes mouvement veut dire que les animats prennent du temps pour accélérer à une vitesse donnée : linéaire ou angulaire. Notez également qu'il n'y a aucun frein une réduction exponentiellement de vitesse du vers la nouvelle valeur.

c. Les paramètres physiques

Les valeurs de m, de C_f, de k_f, de υ, de c_a et de k_a sont fixées pour n'importe quelle expérience. Clairement, les valeurs des paramètres ont un grand effet sur la stratégie de la poursuite ou l'évasion. Le schéma 2 montre la sensibilité des paramètres (vitesse de braquage, vitesse maximale,…).

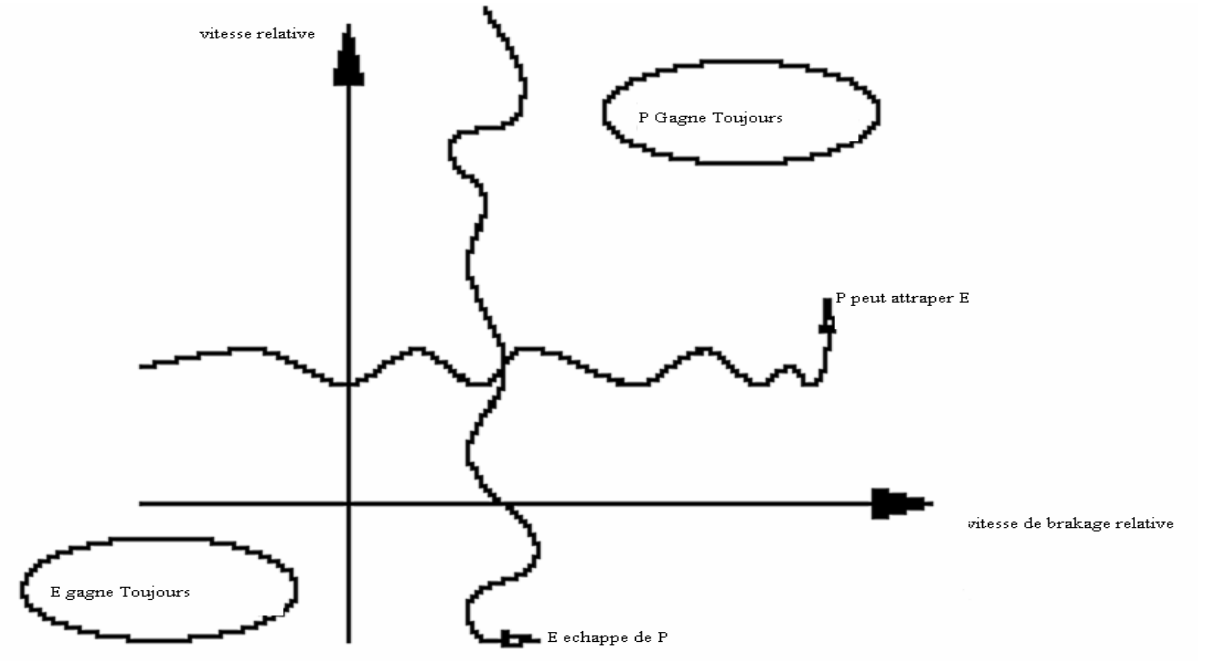


Schéma 2 Scénario de poursuite en fonction Vit relative et Vit braquage relative

Dans la simulation le poursuiveur et le poursuivi ont les mêmes accélérations et vitesses maximum, qui donne un avantage clair au poursuivi. C'est parce que au début de la poursuite la distance suffisante entre le poursuiveur et le poursuivi pour que le poursuivi se tourne vers le poursuiveur avant que le poursuiveur puisse attraper le poursuivi. Pour garder l'équilibre les animats ont des quantités d'énergie différentes avec un plus au poursuiveur, et le poursuivis a une vitesse de braquage maximale un peu supérieure. A la fin, un peu de bruit aléatoire est ajouté sur les valeurs du v_l et du v_r de sorte que le mouvement devient non déterministe.

3.2 Le modèle neuronal du contrôle des entités

Le comportement des animats de la simulation est régi par un modèle neuronal spécial qui est un réseau de neurones récurrent dynamique à temps continu [Beer ,1995], tous le modèle est détaillé ci-après :

a. Le modèle de contrôleur Capteur-Moteur

Un *réseau de neurone*, constitué de *capteur* et d'*effecteurs (de moteur)* est réparti sur la morphologie. Les capteurs recueillent des informations tel que la *détection de contact, ou la mesure d'angle, ou bien la vitesse*, ou encore l'accélération, et aussi des photocapteurs, des capteurs permettant d'écouter, des capteurs permettant de sentir... Cela dépend des comportements que l'on souhaite faire émerger. Le choix sera restreint aux capteurs qui sont nécessaires à accomplir la tâche bien sûr. Les effecteurs permettent d'appliquer des forces entre les différents composants du morphologie, ils sont les homologues des muscles. Pour simplifier l'*antagonisation musculaire* qui se produit très souvent en biomécanique, on laisse les effecteurs pouvoir appliquer des forces négatives.

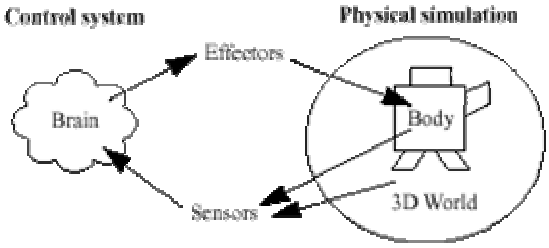


Schéma 3 Le modèle de contrôle

Une fois la créature décrite, morphologie et réseau neuronal, des mécanismes d'évolution sont utilisés pour explorer les différents phénotypes possibles. Ces mécanismes sont basés sur des algorithmes génétiques qui fonctionnent par analogie avec l'évolution génétique de l'ADN que l'on trouve chez les êtres vivants. Les performances des individus sont évaluées, les individus les plus performants sont les seuls à pouvoir subsister. C'est le principe de la sélection naturelle. Ce principe permet de globalement faire évoluer une population, et ainsi du la faire adapter à son environnement.

b. Le modèle neuronal

On emploie des réseaux neurones récurrents dynamique à temps continu. L'activité de neurone est décrite par l'équation: [Rumelhart et McClland, 1986]

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{\forall j} w_{ji}\sigma_j(y_j) + \mathcal{N}_i(t) + \mathcal{I}_i(t)$$

où y_i est l'activité du;τ_i la constante de temps pour le de l'unité i est;$\sigma_j(\xi)$ est une fonction sigmoïdale avec biais (seuil) θ_j ; le w_{ji} est le poids du lien à partir de l'unité j à l'unité i; $N_i(t)$ est le bruit instantané injecté au temps t, et $I_i(t)$ est entrée capteur à l'instant t. Pour chaque unité, les valeurs pour le θ_j et le τ_i, et les limites supérieures et plus inférieures sur la répartition du N_i explicitement spécifiée par un séquence de bits sur le génotype.

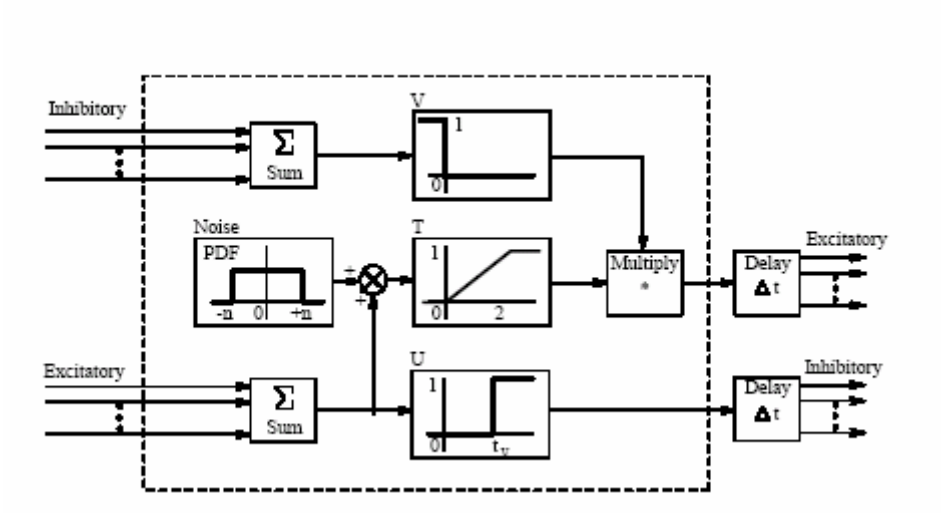


Schéma 4 Le Modèle de neurone

Les valeurs pour le w_{ji} et I_i ont été également génétiquement indiquées, mais pas explicitement. Plutôt, elles ont été engendrées du procédé de morphogénéisation qui sera décrit dans la prochaine section. Le poids w_{ji} du lien reste inchangé au cours de toute l'épreuve de chaque animat, dans ce sens on n'a pas besoin de l'apprentissage du réseau.

c. La vision des animats

On simule une vision plate du monde [Cliff et al, 1996], les capteurs sont placés sur le périmètre du corps circulaire de l'animat avec leur orientation, position relative au corps, et l'angle de l'acceptation tous étant génétiquement spécifiés. La réponse d'un capteur est proportionnelle au pourcentage de son angle d'acceptation qui n'a pas été occupé par le corps de circulaire de l'adversaire. Les équations de projection 2-D ne

sont pas importantes quand on sait qu'il y a seulement un objet dans le monde visuel. Les équations pourraient être résolues ainsi analytiquement, sans recours aux techniques numériques d'approximation mais au bruit aléatoire qui a été ajouté dedans pour s'assurer que les valeurs captées étaient non déterministes et qu'il y avait une limite inférieure sur la résolution c'est à dire des petits signaux pourraient être inondés par bruit.

4. L'approche génétique

Dans cette approche on s'intéresse à donner l'algorithme génétique utilisé, le codage génétique, les opérateurs et enfin touts le déroulement. En premier, on commence par :

Quelques définitions

Gène : Une séquence d'ADN génomique possédant une fonction spécifique. Il correspond à un lieu (locus) sur le génome.
Génome : Ensemble du matériel génétique d'un organisme.
Génotype : La constitution génétique d'un individu. La combinaison de ses allèles à un ou à plusieurs de ses locus donné
Phénotype : Les caractères physiques visibles d'un organisme qui est généré à partir de génotype.
Morphogénéisation: C'est le processus d'obtention ou de passage vers phénotype à partir de génotype.

4.1 Codage avec arbre fractal et morphogénéisation génétique

L'architecture du réseau de neurone, contrôleur sensori-moteur est spécifié par des génotypes de chaîne binaire. En évoluant de tels contrôleurs pour les agents autonomes artificiels nous traitons l'arrangement de la morphologie sensorielle comme partie intrinsèque des spécifications du contrôleur. Le réseau du contrôleur est codé comme une chaîne de bits. On utilise des génotypes de longueur variable pour garder la possibilité de variation du nombre d'unités dans les réseaux, mais on peut employer des génotypes de longueur constante, divisés en un certain nombre de champs. Chaque champ, est préfixé par un bit exprime si ce champs est utilisé pour construire la spécification du contrôleur [Cliff ,1994].

Schéma 5 Exemple d'arbre fractal

Chaque champ contient une séquence de bits qui régissent un paramètre, qui définit un neurone: le neurone à un endroit physique génétiquement spécifié dans le corps circulaire 2D de l'animat; et il y a d'autres paramètres génétiquement affectés qui déterminent la croissance sur le corps d'un arbre fractal d'entrée et d'un arbre fractal de sortie. Si l'arbre d'entrée d'une unité croise l'arbre de sortie d'une autre unité, et certaines autres conditions génétiquement spécifiées sont vérifiées, alors un lien neuronal est établi qui joint l'entrée à la sortie, avec le poids de la liaison affecté par la géométrie de l'intersection des deux arbres (ex. longueur du lien physique multipliée par une constante empirique).Si l'arbre de sortie d'une unité se termine dans une zone centrale sur le corps de l'animat, alors ça devient une unité de sortie de moteur, affectant la valeur gauche ou droite de sortie, selon la géométrie précise des points d'arrêt de croissance. Si l'arbre de l'entrée d'une unité se prolonge au delà du périmètre du corps, alors cette unité devient une unité d'entrée visuelle, avec une position d'orientation , et un angle d'acceptation du photo détecteur correspondant fixé par la géométrie de l'intersection de l'arbre avec le bord de corps. En fin, chaque champ contient une séquence de bits qui peut indiquer l'expression symétrique, si un neurone est exprimé symétriquement, alors une copie est créé en reflétant les arbres du corps et de l'entrée et de sortie du neurone autour de l'axe longitudinal de l'animat.[Yao, 1997]

a. Vivacité du phénotype

Ce schéma de codage s'est avéré offrir une puissance considérable. Il permet que quelques unités le réseau soient cacher (cf. inter neurones) d'autres à être des neurones capteurs, et d'autres soient des neurones des sorties de moteur. Mais le plus intéressant, il est possible qu'une unité soit capteur et sortie de moteur. Ces possibilités couplées qui'est la faculté relative de produire des conceptions bilatéralement symétriques, c'est-à-dire, que les génotypes aléatoires codent souvent des contrôleurs simples mais très efficaces comme des véhicules de Braitenberg [Braitenberg 1984]. Souvent, les conceptions aléatoires peuvent n'avoir aucune sortie de moteur, aucune entrée sensorielle, ou aucun lien entre les capteurs et les moteurs.

Pour cette raison, on nous employons « l'eugénique » d'animat pour augmenter l'efficacité, en produisant de la population aléatoire initiale des génotypes, ou en

croisant un nouvel individu de deux parents, un contrôle de viabilité est fait à des individus avec des nombres insuffisants des capteurs ou des moteurs sont rejetés immédiatement. Alors le génome est exprimé pour donner le contrôleur et un collecteur de miettes (garbage collector) supprime toutes les unités non connectées. Si, après collection de miettes, le réseau est encore viable, il est ajouté à la population. Autrement, le génotype est jeté et un autre est produit jusqu'à ce qu'un génotype viable résulte.

b. ADN « Junk »

Un aspect intéressant, employer ce schéma de codage est dans la pratique, qu'en général il y a des séquences de bits sur le génotype qui constitue l' « ADN *JUNK »(ADN ORDURE)*, du fait ils ne contribuent pas au phénotype à cause du bit de participation au phénotype est réglé à « off ».Cependant, il faut être prudent au sujet de l'utilisation du terme "ADN *JUNK*", le croisement ou la mutation suivante pourrait mener à rendre une unité non exprimée ;exprimée et l'inverse , même déconnecter une unité déjà connectée. Ainsi, il est préférable d'utiliser le mot « silencieux »plutôt que le mot « junk ».Un effet pratique de l'ADN silencieux dans les génomes est qu'une mutation dans un ordre silencieux n'a aucun effet immédiat. Pour parer ceci, nous employons les taux de mutation qui seraient considérés très hauts dans des codages plus standard où chaque bit compte

4.2 Le déroulement de l'algorithme génétique

Notre algorithme génétique GA fait évoluer une population des poursuiveurs et une population de poursuivis. Les deux populations sont de la même taille. Nous employons un GA spatialement distribué, où différents génotypes dans chaque population sont étendus pour occuper des positions discrètes sur une grille. Les individus reproduirent seulement avec les voisins, et la progéniture remplaceront tous les individus moins adaptés. La grille a une topologie toroïdale donc il n' y a aucun effet de bord.

Le GA procède en des générations discrètes. Dans la toute première génération, chaque individu est testé contre un ensemble d'individus aléatoirement choisis de la population opposée. Dans toutes les générations suivantes, des individus sont examinés contre le meilleur individu dans la population opposée sur la génération précédente « Last Elite Opponent » (LEO le meilleur opposant de la génération précédente).[Miller et Cliff, 1994]

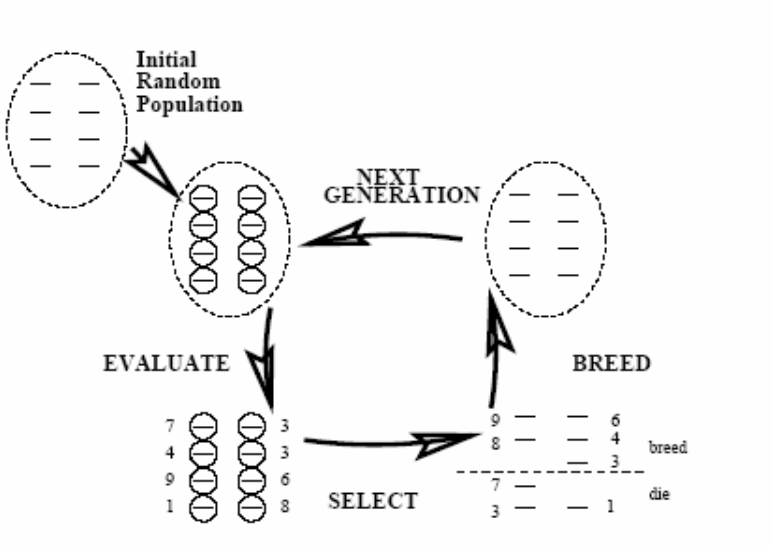


Schéma 6 Le cycle de l'algorithme génétique

Dans chaque essai, le génotype est donné un certain nombre d'épreuves qui sont différents concours de poursuite. Les conditions initiales pour les épreuves sont changées en utilisant des précautions statistiques standard. La qualité (fitness) finale de l'individu a été calculée par la moyenne des fonctions objectives de l'ensemble des d'épreuves. La fonction objective d'un poursuivi est simplement la durée avant qu'il soit attrapé par le poursuiveur. La fonction objective du poursuiveur est légèrement plus complexe, le poursuiveur reçoit des points bonus chaque fois qu'il approche au poursuivi, et si le poursuiveur attrape le poursuivi il reçoit un bonus plus grand est qui dépend du temps de la poursuite. Chaque épreuve, le poursuivant et le poursuivi commencent à leurs positions et orientations d'initiale, avec toutes les activités dans leurs contrôleurs neuronaux réglés à zéro. Une fois que l'épreuve a commencé elle continue jusqu'à ce que les animats se heurtent, ou tous les deux soient en manque d'énergie, ou un délai fixe soit atteint.

a. La reproduction (Breeding)

Une fois que tous les individus dans les deux populations ont été évalués, deux nouvelles populations sont reproduites. La reproduction emploie les opérateurs standard de GA de la mutation et du croisement, plus un opérateur de duplication discuté plus loin dans la prochaine section. Pour reproduire de nouveaux un parent de l'individu deux individus sont choisis en utilisant le choix basé par rang probabilistique. Un des deux parents est choisi au hasard et le processus de copier son génotype dans le génotype de l'enfant débute. Une fois que chaque bit est copié un nombre aléatoire est produit d'une distribution uniforme. Si c'est inférieur une à certaine valeur seuil alors l'indicateur copiant est commute à la position correspondante sur l'autre parent. Le nombre de croisements dans la reproduction de n'importe quel enfant suit ainsi une distribution de Poisson, là sera toujours une probabilité finie que la reproduction soit asexuel. La mutation est également appliquée sur une base sage de bit avec une valeur seuil différente. Ainsi le nombre de mutations par reproduction suit également une distribution de Poisson.

b. Un opérateur génétique spécial « *Duplication* »

La problématique :

Le codage génétique et la technique de morphogenèse utilisés, avec les contrôles de viabilité et la collection de miette engendrent des individus avec des architectures neurales très simples semblables aux véhicules de Braitenberg, avec un arrangement symétrique de quelques capteurs reliées aux moteurs. Très souvent, n'importe quelle amélioration n'augmente pas le nombre de neurone de l'animat. Alors, même après des centaines de générations, l'amélioration des résultats était souvent seulement un petit pourcentage par rapport à l'architecture initiale. On explique ce phénomène, par le fait que ces petits contrôleurs neuronaux de modèle de Braitenberg ont typiquement des frontières épistatiques très élevées autour d'elles dans l'espace de génotype. Si on ajoute un ou plusieurs neurones supplémentaires au hasard au contrôleur neuronal on a presque toujours une dégradation significative de la qualité de l'animat. Les contrôleurs initiaux étaient si petits qu'en ajoutant de nouveaux neurones est typiquement un impact négatif énorme sur l'exécution du réseau.
Pour remédier à ce problème, il y a un autre opérateur génétique dans la nature :

Définition :

Beaucoup de systèmes artificiels évolutionnaires utilisent le croisement et la mutation, la duplication est moins largement répandue. Dans notre exécution la duplication prendrait un champ génétique et le copierait dans un autre champ sur le génotype. Si le champ avait son bit d'expression réglé à « 1 » actif, le réseau résultant de contrôleur aurait les neurones supplémentaires, mais ils seraient presque, mais identique aux unités dans le réseau courant, et ainsi serait peu susceptible d'avoir un effet négatif sur le comportement global. Une fois que la duplication était utilisée, on constate que le nombre d'unités augmenterait souvent de manière significative au cours d'exécution.

Remarque : L'opérateur de duplication est très utilisé dans les systèmes de programmation génétique, en abrégé GP, définis par [Koza, 1992].

5. Quelques résultats de simulation

Après avoir détaillé les méthodes et techniques utilisées dans la simulation de poursuite, on va donner quelques résultats. La première constatation est, le temps prohibitif des exécutions des dizaines d'heures processeur.

5.1 L'évolution des contrôleurs d'animat

Dans le tableau suivant, on voit les résultats du processus de morphogénéisation et de l'algorithme génétique d'évolution. Les résultats donnés sont pour les

générations 0, 200, et 999. Les schémas sont des vues de dessus de l'animat pour les deux meilleurs animat de chaque espèce des générations citées avant.
On remarque, que d'une génération à l'autre les animats croient en complexité d'architecture, nombre de capteurs, nombre de neurones, et s'améliore en comportement.

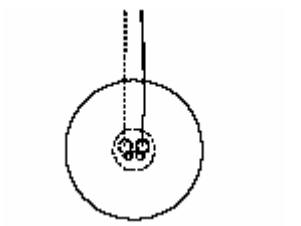
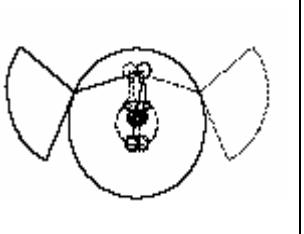
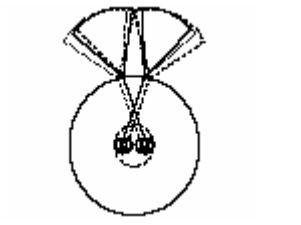
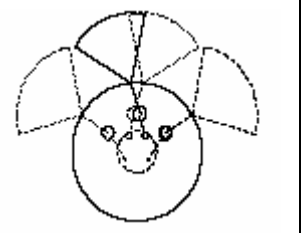
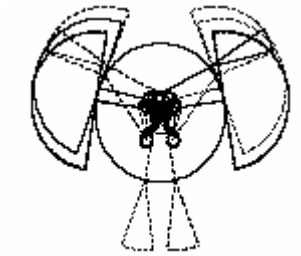
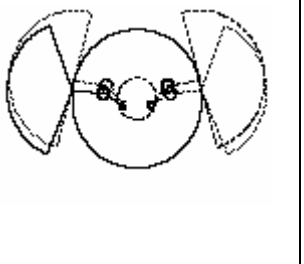
	Meilleur poursuiveur	**Meilleur poursuivi**
Génération 0		
Génération 200		
Génération 999		

Tableau des évolutions des entités au cours des générations

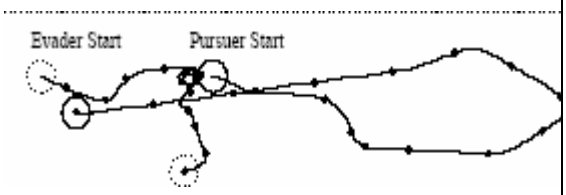
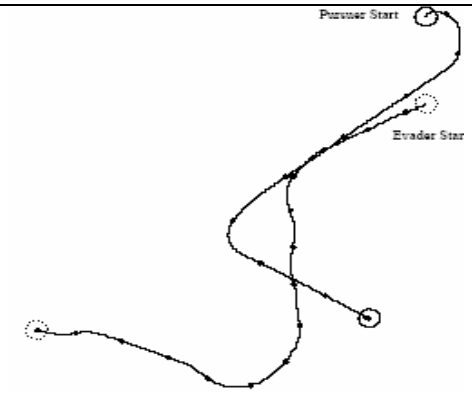
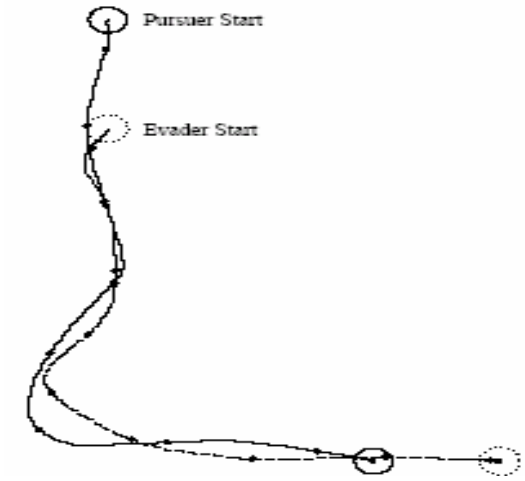
Explication du tableau

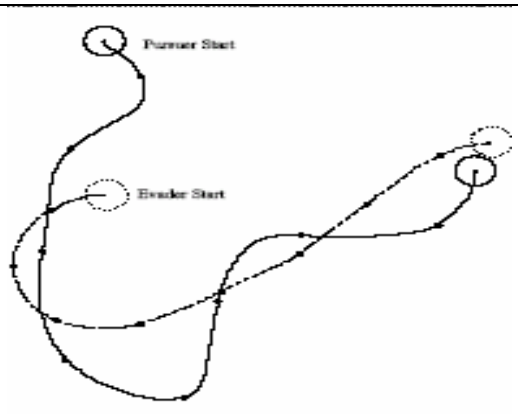
Chaque schéma est une vue de haut de l'animat, la face frontale est celle de haut, le grand cercle représente l'étendu du corps de l'animat, les cercles discontinus intérieurs représentent les cordes spinales responsables d'asservissement des roues, les autres cercles intérieurs représentent des neurones ; les petits cercles représentent les synapses, les grands représentent des cellules du corps de l'animat et les lignes qui les relient les interconnexions neuronales. Les points d'intersection entre les lignes et l'étendu du corps représentent des capteurs avec un angle d'acceptation (angle de vision) génétiquement spécifiée dans le génotype. Par exemple, le meilleur

poursuiveur de la génération 999, contient dix capteurs dont deux parmi ceux sont positionnés à l'arrière de l'animat.

5.2 Les résultats de la simulation de la poursuite

Dans ce tableau explicatif, on va voir les scénarios de poursuite, entre le meilleur poursuiveur et le meilleur poursuivi des génération 0, 200, et 999, en plus un scénario de poursuite entre le meilleur poursuiveur de la génération 999 et le meilleur poursuivi de la génération 200, pour voir clairement l'évolution des capacités des animats entre générations.

La génération	Le scénario de la poursuite	Explication
Génération P0 vs. E0		*Génération 0 :* Dans ce scénario le poursuiveur et le poursuivi ont des contrôleurs opérationnels, c'est-à-dire peuvent faire seulement des mouvements mais la poursuite (resp. l'évasion) c'est tout une autre chose.
Génération P200 vs. E200		*Génération 200 :* Dans ce scénario le poursuiveur et le poursuivi ont des contrôleurs assez performants, mais le poursuiveur reste incapable d'attraper le poursuivi de la même génération, à cause de l'utilisation abusive de sa vitesse maximale, qui fait consommer son énergie limitée dans un temps réduit. On remarque de même que le poursuivi sait très bien utiliser ses capacités pour faire des braquages satisfaisants.
Génération P999vs.E999		*Génération 999 :* Dans ce scénario le poursuiveur et le poursuivi ont des contrôleurs très performants, mais le poursuiveur reste incapable d'attraper le poursuivi de la même génération (999), avec une poursuite parfaitement optimisée c'est-à-dire on a, des braquages exacts, une utilisation d'énergie excellente, un réflexe de perception très performant, et c'est la même chose pour le poursuivi. Maintenant, on voit pourquoi le meilleur léopard n'attrape pas la meilleure gazelle, mais une petite ou faible gazelle !

Génération P999 vs. E200		*Génération P999 vs. E200 :* Dans ce scénario le poursuiveur et le poursuivi ont des contrôleurs assez performants, mais la différence de capacité est flagrante le poursuiveur de la génération 999 peut prévoir les mouvements du poursuivi de la génération 200 et l'attraper en un temps très réduit avec des perturbations marginales du comportement du poursuivi[Cliff et Miller, 1995]. Maintenant, on voit clairement que les contrôleurs de la génération 999 sont très performant même s'ils ne peuvent pas attraper un poursuivi de la génération 999.

6. Critiques et proposition d'extensions

Après l'analyse du problème et les simulations effectuées, on a constaté qu'il souffre de quelques inconvénients, ils sont cités en des critiques :

6.1 Les critiques

- Le monde de simulation ne contient pas des obstacles, ce qui rend la simulation incapable de prédire le comportement des animats dans le cas où ça existe [Sims, 1994].
- La morphologie des entités est très simple : deux roues, et quelques capteurs
- Les poursuiveurs des dernières générations trouvent des problèmes dans la poursuite des poursuivis des générations très anciennes « *Conflits d'age* ».
- La dynamique physique bidimensionnelle utilisée rend la simulation très simpliste ni projections, ni positions tridimensionnelle [Stork et al, 1992].
- La non-existence des champs pour l'évolution de la communication, dans le génotype, ce qui pose problème pour l'extension du modèle à des simulations avec plusieurs entités.
- Les durées énormes des simulations, sont de l'ordre de dizaines d'heures.
- Un des problèmes critiques, est celui de l'évaluation de l'adaptabilité des animats entre générations,à cause du manque de « Fitness Landscap » (échelle de fonction objectif) -voir le chapitre des algorithmes génétiques-, une des techniques utilisées on trouve LEO entre plusieurs générations [Cliff et Miller, 1995].

6.2 Les extensions du problème de la poursuite

Le problème de la poursuite reste un paradigme très puissant pour des modélisations, après avoir vu quelques critiques sur le modèle de simulation et les techniques utilisées. Il nous semble intéressant de faire quelques extensions structurelles et organisationnelles,

6.2.1 Extensions structurelles

Les extensions structurelles concernent les caractéristiques des animats, pour gagner en puissance de modélisation réelle, c'est-à-dire dans les robots réels. Voici quelques extensions :

a. Les dépôts d'énergie

La simulation est très liée à l'énergie des animats, une extension souhaitable, est de mettre de l'énergie dans le monde de simulation, cette extension est inspirée de la nature les sources de nourriture. Ce changement va avoir un impact sur les capteurs donc il faut avoir un type spécial pour trouver l'énergie, comme modification possible, on va avoir un nouveau champs sur génotype qui fait évoluer des capteurs de nourriture.

b. La communication

Les animats de la simulation n'ont pas un moyen de communication, ce qui n'est pas un problème pour des simulations « un poursuiveur vs. un poursuivi », mais le problème commence à apparaître quand on fait une extension « plusieurs poursuiveurs vs. plusieurs poursuivis ». Pour implémenter cette extension, il suffit d'ajouter un champ dans génotype, qui fait créer des liens pour la communication, sous bien sur un langage défini.

c. Codage génétique

Le codage génétique à base d'arbres fractals, fait croître le temps d'exécution de façon significatif, alors préférable d'essayer de voir d'autres codages comme le codage cellulaire de [Gruau, 1992], ou matriciel de [Kitano, 1990], ou grammatical de [Kodjabachian et Myer, 1998].

d. Les limites morphologiques

Dans la pratique, on peut avoir des robots très complexes avec plusieurs roues et plusieurs capteurs, une extension pratique, c'est de permettre au processus de morphogénéisation de générer des animats plus complexes, pour utiliser les contrôleurs engendrées dans des applications réelles de robotique [Miller et Freyd, 1993].

e. La dynamique physique

La dynamique des animats de la simulation est simple, c'est un mouvement newtonien, une extension possible est de tester des équations de mouvement plus

générales, comme des équations différentielles partielles qui enveloppent l'allure du mouvement.

6.2.2 Extensions organisationnelles

Les extensions organisationnelles se basent sur le paysage (monde) de la poursuite, c'est le plan de simulation, et les entités, voici deux extensions possibles :

a. Extensions du monde

Notre monde est vide et bidimensionnel, on fait remplir le monde par des obstacles et étendre le monde à trois dimensions. Cette extension peut facilement augmenter le temps d'exécution des simulations et la structure du contrôleur neuronal de l'animat.

b. Monde avec obstacles

Cette extension va influée sur la fonction objective des deux types d'entités, on va avoir une fonction multi objectives. C'est la somme de la première fonction objective avec le nombre d'obstacles que l'animat a touché, de cette façon la coévolution va produire des poursuiveurs qui évitent les obstacles et attrapent les poursuivis, et pour les poursuivis échappent aux poursuiveurs.

c. Monde tridimensionnel

Cette extension va influée sur l'équation de mouvement, on doit faire des projections (x,y,z) de l'équation pour trouver les composantes tridimensionnelles de v_l, v_r et de l'accélération, dans ce cas on va avoir (f_x, f_y, f_z) composantes de f la position et $(\alpha_x,\alpha_y,\alpha_z)$les composantes de α la position angulaire.

d. Extension à base du leader

Une extension habituelle [Grefenstette, 1992], est de faire attraper un poursuivi par plusieurs poursuiveurs contrôlés par un seul maître, pour faire ce là il suffit de :

- Définir dans le génotype des poursuiveurs un champ pour la communication
- Définir le type de communication directe ou via l'environnement.
- Définir les concepts et les informations à communiquer,
  - La position relative du poursuivi,
  - La vitesse et la direction relatives du poursuivi,
  - La position du maître,
  - Les points de rendez-vous,
- Les règles de gestion des conflits entre poursuiveur.
- Définir c'est quoi la capture collective ? en peut prendre par exemple une il suffit de bloquer le poursuivi de telle sorte qu'il ne peut effectuer des déplacements.

Cette extension souffre d'un inconvénient majeur, qui est, le non-respect de l'équilibre entre espèces nommé « équivalence de chances », exigé dans n'importe quel système coévolutionnaire, si ce n'est pas le cas on va pas bénéficier de la puissance de la coévolution qui a généré des poursuiveurs très performants et des poursuivis de mêmes. Le futur de l'autre espèce « les poursuivis » est prédéterminé - une disparition instantanée ; très proche à celle des dinosaures – donc on va pas avoir lieu à une coévolution entre espèces.[Werner et Dyer, 1993]

7. Conclusion

Le problème de poursuite de Cliff&Miller, est un classique de la robotique adaptative, mais une robotique collective s'impose dans la pratique, après avoir vu quelques extensions structurelles et organisationnelles, on voit qu'il est impératif de faire une formalisation du problème pour la généraliser dans un contexte d'environnement collective, et c'est le but du chapitre de la proposition.

Les Systèmes multi agents

1 Introduction

La communauté de l’Intelligence Artificielle Distribuée (IAD) n’a pas à nos jours donnée une définition commune d’un système multi-agents (SMA), c’est peut être parce qu'il est difficile de définir la notion d'agent.
Poser la question qu’est ce qu’un agent pour ceux qui travaillent en IAD est aussi difficile que poser la question qu’est ce que l’intelligence pour la communauté de l’IA [Hewitt, 1991]. En effet, la notion d’agent diffère selon l’axe de recherche envisagé. Ferber [Ferber, 1995] a préféré employer le terme kénétique qui regroupe l’ensemble des théories et réalisations dans ce domaine. Notre points d’intérêt dans ce domaine est de ce focaliser sur la communication pour résoudre le problème de la poursuite de Cliff & Miller étendu dans le chapitre suivant (l’approche COCAGE).

Au de là des multiples définitions, on distingue dans la littérature deux définitions d’agents, la première est dite "faible" et évidente, et la deuxième est dite "forte".

2 Agent et systèmes multi-agents

2.1 Définition faible de la notion d'agent

Un agent est une entité matérielle ou logicielle qui a les propriétés suivantes [Wooldrige et Jennings,1994] :

- *Autonomie* : les agents sont doués d'autonomie, cela signifie qu'ils ne sont pas dirigés par des commandes venant de l'utilisateur ou d'un autre agent [Ferber, 1995].
- *Aptitude sociale* : les agents interagissent avec d'autres agents via un langage de communication pour agent [Chaib,1994].
- *Perception* : les agents ont les capacités de percevoir leur environnement (mais d'une manière limitée).
- *Engagement* : les agents ne contentent pas seulement de répondre à leur environnement, ils peuvent, suivant leur but, prendre des initiatives [Della,2000].

2.2 Définition forte de la notion d'agent

Les prétendants de cette notion (particulièrement ceux qui travaillaient en IA), veulent implanter, en plus des propriétés citées ci-dessus, les concepts inhérents à l'homme. Donc l'agent est caractérisé avec des notions désignant des états mentaux : la connaissance, la croyance et l'intention [SHOHAM, 1993], l'émergence [Ferber, 1995].

L'agent est donc une entité physique ou logicielle :

- qui est capable d'agir dans un environnement (Activité) ;
- qui peut communiquer directement avec d'autres agents ;
- qui est mue par un ensemble d'objectifs propres (sous forme d'objectifs individuels ou d'une fonction de satisfaction) ;
- qui possède des ressources ;
- qui est capable de percevoir son environnement ;
- qui ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune) ;
- qui possède des compétences (services) qu'elle peut offrir aux autres agents ;
- qui a un comportement tendant à satisfaire ses objectifs, en tenant compte d'une part des ressources et des compétences dont elle dispose, et d'autre part de ses perceptions, de ses présentations et des communications qu'elle reçoit.

2.3 Autres attributs d'agents

On trouve aussi, en plus des attributs définit en haut, d'autres attributs parmi lesquels : [Galliers, 1988]

- *sincérité* : qui indique qu'un agent ne doit pas communiquer de fausses informations ;
- *la rationalité* : qui indique le fait qu'un agent agisse pour atteindre ses buts ;

- *la mobilité* : qui est la capacité qu'a un agent à se mouvoir dans un réseau.
- *La réactivité* : l'agent peut répondre aux changements de l'environnement qu'il perçoit.

2.4 Modèle d'agents

On trouve plusieurs types d'agents dans la littérature, et selon leur niveau d'intelligence et leur domaine d'applications, parmi ces agents, nous pouvons citer :

- Les agents communicants qui possèdent des capacités de communication complètes, mais leur capacité de raisonnement est liée au domaine d'activité. [Della,2000]
- Les agents de traitement dont le rôle sont d'effectuer des transformations qui sont des traitements ou des calculs sur des flots de données en entrée pour produire des flots de données en sortie. Il assure l'exécution d'un algorithme de complexité quelconque. [Attoui,1997]
- Les agents naturels qui sont des agents dotés de possibilités d'auto apprentissage (faculté d'augmentation de sa base de connaissance) par ajout de règles de fonction de l'évolution de son environnement et des événements qui peuvent se produire. [Attoui,1997]
- Les agents spécialistes ont une compétence précise et disposent d'une représentation partielle de leur environnement. Dans un système d'agents spécialistes, il existe des moyens d'affectation des tâches entre agents. [Vincent, 1993]

Mais, il y a deux conceptions qui ont données lieu à deux écoles de pensée différente, ces conceptions ont un grand effet dans la recherche sur les SMA's, ce sont les agents cognitifs et réactifs.

2.4.1 Agents cognitifs (école cognitive ou sociale)

Elle est la plus représenté en IAD, car elle trouve son origine dans la volonté de faire communiquer et coopérer des systèmes experts classiques. Dans ce cadre, un système multi-agents est composé d'un petit nombre d'agents ''intelligent'', chaque agent dispose d'une base de connaissance [Ferber, 1995].
Cette classe d'agent tend à simuler les aptitudes (comportement) humaines.

2.4.1.1 caractéristique d'un agent cognitif

On dit que les agents cognitifs sont :

- *Intentionnels* : c'est-à-dire qu'ils possèdent des buts et des plans explicites leurs permettant d'accomplir leurs buts. [Ferber, 1995].
- *Rationnels* : le mot provient de la raison, et signifie que si un agent sait qu'une de ces actions lui permet d'atteindre un de ses buts, il la sélectionne. [Khoualdi, 1994]
- *Adaptatifs* : c'est la flexibilité, c'est-à-dire qu'un agent est autonome, sociable (interaction avec d'autres agents), réactif (perception de son environnement et réaction), proactif (il peut prendre des initiatives de manière à stimuler l'environnement pour la réalisation de son but) [Wooldrige et Jennings, 1995].

- *Intelligents* : c'est l'ensemble de fonctions mentales ayant pour objet la connaissance conceptuelle et rationnelle. Un agent intelligent est un agent cognitif, rationnel, intentionnel et adaptatif.

2.4.1.2 architecture d'un agent cognitif

A. *Structure interne :* La structure interne d'un agent cognitif (Fig. 1.) comprend les éléments suivants : les croyances, les contrôles, l'expertise, les connaissances de communication et de coordination. [Baujard,1992]

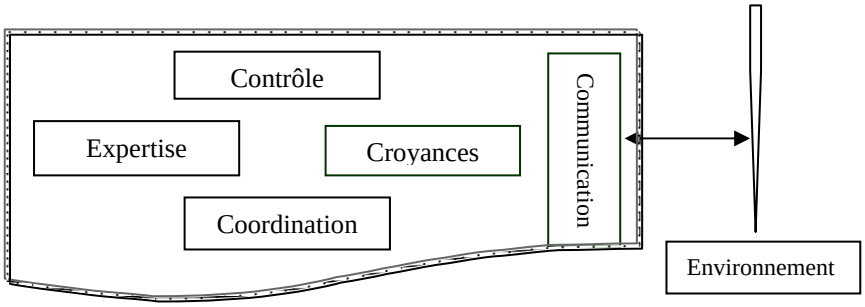


Fig. 1. Structure interne d'un agent

- Les croyances : les croyances d'un agent sont des connaissances, pas nécessairement objectives, sur soi (méta connaissance) qui est caractérisé par son nom, son identificateur, son adresse, ses objectifs à attendre et son état interne, et sur les autres appelés ses accointances, qui est une représentation partielle (adresses des accointances et leurs buts).
- L'expertise : c'est les aptitudes de résolution de l'agent dans un domaine. Elle représente la connaissance sur la résolution d'un problème. [Ferber, 1989]
- La coopération : Elle concerne les connaissances sur les modes d'interaction entre agents. Elle est liée à la communication inter-agents.
- Le contrôle :l'agent doit déterminer l'action à entreprendre en fonction des informations, des connaissances, des intentions et des buts dont il dispose pour assurer une fonction tout en conservant son intégrité structurelle.
- La communication : Les communications sont la base des interactions entre les agents dans un système multi-agents. Car sans communication l'agent n'est qu'un individu isolé, sourd et muet.

La communication est l'un des aspects les plus important en IAD qui permet l'échange d'informations entre agents, la formulation de requête, …etc. Plus un système est important, plus il y a un besoin accru de coopération et de communication entre agents.[Vincent, 1993]

B. *Fonctionnement :* Les agents cognitifs fonctionnent en respectant les trois étapes suivantes (Fig. 2.) :

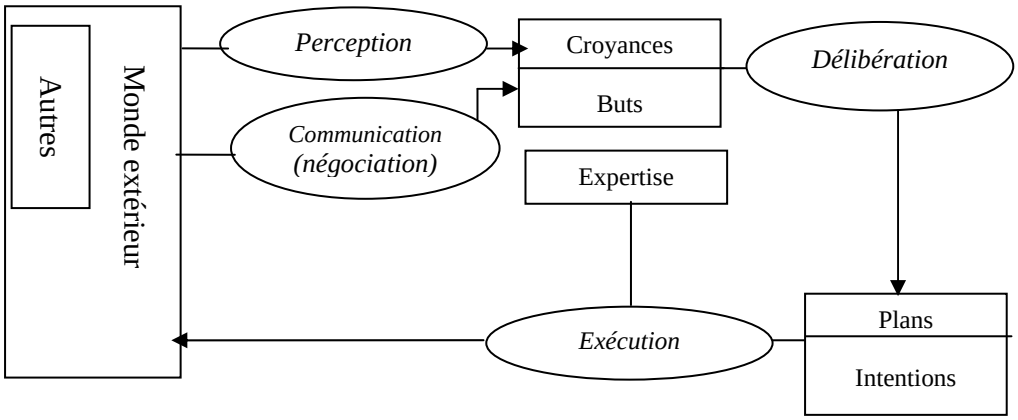


Fig. 2. Fonctionnement d'un agent cognitif.

- ***Perception*** : elle fournit les capacités d'entrée nécessaires à l'agent, et de pouvoir classer et distinguer les états du monde. Elle comprend le savoir initial de l'agent, la perception de soi et des autres et la communication avec les autres. C'est à dire que l'agent détecte la présence de toute entité initialement inconnue dans le système [Cha96] (par exemple d'autres agents, des obstacles …etc.), il reçoit aussi des messages des autres.

- ***Délibération*** **: c'est la partie la plus essentielle d'un agent, c'est la que s'élaborent les objectifs, les prises de décision, la détermination de plan et la séquence d'actions qui permet de satisfaire les buts (planification). [Ferber, 1995]**

- ***Exécution*** **: Fournit à l'agent les capacités nécessaires de sortie (output), elle permet d'agir en effectuant certaines actions. [Della,2000]**

2.4.2 **Agents réactifs** (école réactive ou biologique)

Cette tendance prétend qu'il n'est pas nécessairement que les agents soient intelligents individuellement pour que le système ait un comportement global intelligent, ainsi on fait coopérer un grand nombre d'agent de faible granularité pour aboutir aux objectifs fixés, ces agents ont un comportement élémentaire de type stimulus / réponse.

Les premiers travaux relatifs à cette approche ont été réalisés en 1986 par Brooks [Brooks,1986], qui a fait coopérer plusieurs micro-robots de petite taille pour réaliser seulement une tâche donnée, et selon lui, elle est plus efficace que de travailler avec un seul gros robot.

La plate-forme MERING a été utilisée pour la conception de SMA réactif.[Ferber et carl, 1991]

2.4.1.1 architecture d'un agent réactif

- *Structure Interne : Un* agent réactif peut être modélisé par un simple objet doté d'un comportement et d'un moyen de communication avec les autres agents. Il n'a pas de connaissances sur les autres, ni de capacités à raisonner sur les messages qu'il reçoit, ni même de développer des stratégies de contrôle. Ils sont donc orientés plutôt comportement que connaissance.

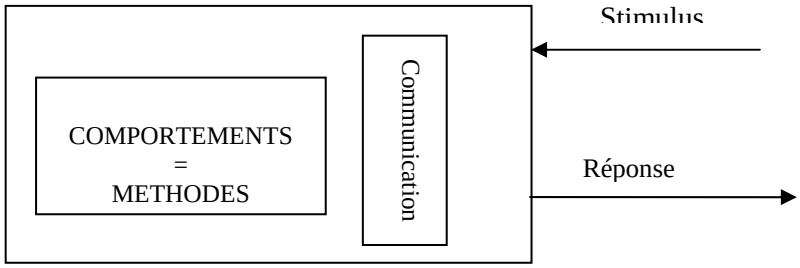


Fig. 3. Architecture d'un agent réactif

- *Fonctionnement :* Ces agents aux comportements élémentaires sont susceptibles d'interagir qu'aux stimuli observés, leurs interactions peuvent émergées de l'intelligence.

2.5 Les Systèmes multi agents

2.5.1 Définitions

Selon Ferber et Ghallab [Ferber et Ghallab, 1988], «un système multi-agents (SMA) est une communauté d'agents travaillant en commun selon des modes parfois complexes de coopération, de conflit et de concurrence pour aboutir à un objectif global : la résolution d'un problème, l'établissement d'un diagnostic …etc. ».
Le principe est que si un agent détient l'expertise qui lui permet de résoudre le problème en entier alors il le résout selon les données et les ressources dont il dispose, dans le cas contraire, il use de toute la puissance de ses moyens de communication pour arriver à une satisfaction globale. [Haton et al, 1991]

Selon Jennings, Wooldridge et Sycara [Jennings et al, 1998], un SMA peut être défini comme un ensemble de « résolveurs » appelés agents qui travaillent ensemble pour résoudre des problèmes dont aucun n'a les capacités ou les connaissances individuelles pour les résoudre totalement.

Le principe adopté serait qu'aucun agent n'ait les informations ou les capacités suffisantes pour résoudre le problème : il a une vue limitée, il n'y a pas un système de contrôle global, les données sont décentralisées et la résolution est asynchrone.

2.5.2 Les modèles de SMA

Pour faire communiquer des entités informatiques, les systèmes d'IAD ont eu recours à différentes approches : les tableaux noirs, les acteurs et les systèmes physiquement distribués

2.5.2.1 Les architectures à base de tableau noir (BlackBoard)

L'architecture à base de tableau noir est l'une des plus utilisé dans les systèmes multi-agents cognitifs, originalement développée dans le cadre de l'IA pour la reconnaissance de la parole avec un système, baptisé HEARSAY. [Erman et al, 1980]

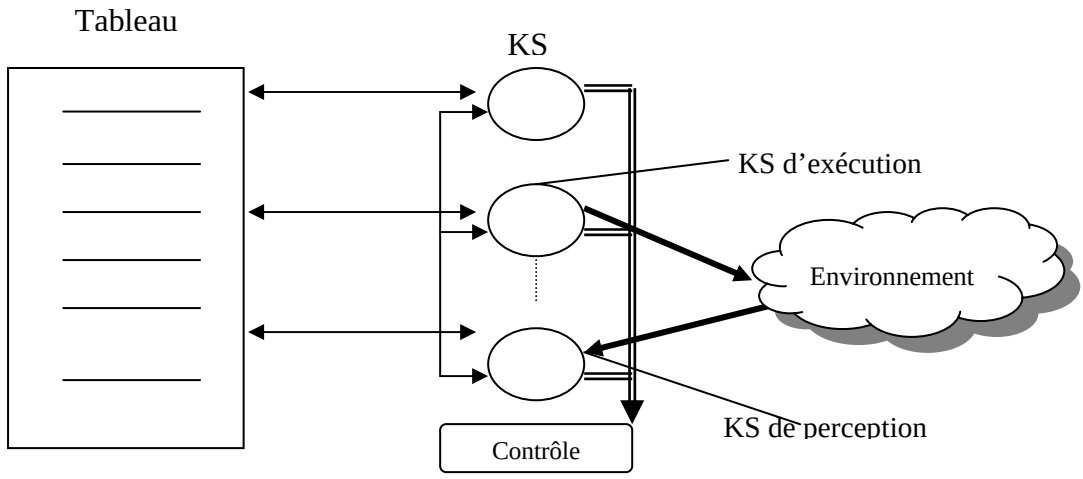


Fig. 4. Une architecture de système à base de tableau noir.

Le modèle à tableau noir est fondé sur un découpage en modules indépendants qui ne communiquent directement aucune information, mais qui interagissent en partageant des informations, ces modules appelés sources de connaissance ou KS (pour Knewledge Sources) travaillent sur un espace qui comprend tous les éléments nécessaires à la résolution d'un problème. L'architecture de Fig. 4 comprend trois sous systèmes [Ferber, 1995] :

- **La source de connaissance KS ;**
- **La base partagée (le tableau) : qui comprend les états partiels d'un problème en cours de résolution, les hypothèses et les résultats intermédiaires. D'une manière générale toutes les informations que s'échanges les KS ;**

- **Un dispositif de contrôle qui gère les conflits d'accès entre les KS, ces dernières interviennent sans être déclenché par un système de contrôle centralisé.**

Le problème de contrôle dans un tableau noir [Ferber, 1995] revient à essayer de savoir ce qu'il convient de faire ensuite, c'est à dire déterminer quelle KS doit être déclenchée. Au début, avec HearsayII, le contrôle était câblé au sein d'une procédure, puis il fut donnée sous forme d'un ensemble de règles, mais ce n'est qu'avec le système BB1 de Hayes-Roth que le contrôle a acquit ses lettres de noblesse, en considérant que le contrôle est un problème important pour qu'il dispose de son propre tableau [Hay85].

Les avantages de ce type d'architecture sont indéniables : une remarquable souplesse pour décrire les modules et articuler leur fonctionnement, il est possible d'implémenter n'importe quelle structure d'agent en terme d'éléments de tableau et de KS. Son principal inconvénient est du fait que son contrôle très centralisé (le module de contrôle qui ordonne l'ordre dans lequel les KS seront activés) et de leurs manques de mémoire locale.

Le tableau noir se présente comme une sorte de méta-architecture c'est à dire une architecture pour implémenter des architectures.

2.5.2.2 Les systèmes à acteurs

Un acteur est une entité informatique qui se compose de deux parties : une structure qui compose l'adresse de tous les acteurs qu'il connaît (ces accointances) et qu'il peut les envoyés des messages, et une partie active, le script, qui décrit son comportement lors de la réception un message (Fig. 5.)[Hewitt et al, 1973][AGHA,1986].
Le comportement de chaque acteur, qui s'exécute indépendamment, et en parallèle avec les autres, se résume à un ensemble d'action réduit : envoyer des messages, créer des acteurs et modifier son état, et c'est suffisant pour pouvoir exprimer n'importe quel calcule parallèle.

La communication entre acteurs s'effectue par des envois des massages asynchrones. [Jennings et al, 1998]

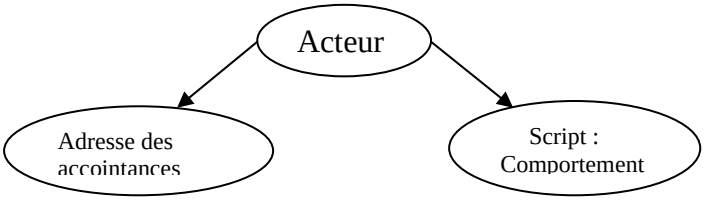


Fig. 5. Modèle d'acteur.

2.5.2.3 Les systèmes physiquement distribués

Les systèmes proposés en planification multi-agents centralisée présentent des options intéressantes pour la résolution des conflits et des convergences vers une solution globale. Cependant, ils ont tous les mêmes inconvénients liés à la centralisation du contrôle au niveau d'un seul agent.
Les différents agents travaillant d'une façon concurrentielle dans le réseau doivent envoyer leurs actions au coordinateur, afin qu'il résolve les conflits éventuels et gère leurs interactions.

La charge de communication est très importante : pour cette raison les chercheurs ont pensé à des systèmes à planification distribuée. Ces systèmes sont caractérisés par un ensemble d'agents complexes, dotés de grandes capacités de raisonnement, et distribués sur des sites géographiquement répartis. Ceci nécessite la constitution de mécanismes de coordination très élaborés entre agents.

3 Sociétés d'Agents

Dans les applications multi-agents, les entités sont autonomes et chacune de ces entités ont des buts différents. Ceci peut conduire à une incohérence du système lors de la résolution du problème, les agents sont organisés en société régie par des comportements globaux structurés. Ils sont structurés grâce aux théories sociales pour la résolution d'un problème posé.

Dans cette partie, nous nous intéressons au comportement global d'une société d'agents et aux théories sociales requises pour la résolution de problèmes dans un univers multi-agents à savoir l'organisation, la coopération entre les agents, les moyens de communication, la résolution des éventuels conflits dus à la diversité de buts des agents faisant intervenir la coordination et la négociation et enfin l'émergence d'intelligence dans une interaction entre agents réactifs.

3.1 L'Organisation Sociale

L'organisation sociale d'un système multi-agents est la manière dont le groupe est constitue, a un instant donné, pour pouvoir fonctionner. Elle décrit l'ensemble des composants fonctionnels du système, leurs natures, leurs responsabilités et leurs besoins en ressources ainsi que les liens de communication entre les agents. Cette organisation peut être statique ou dynamique.
Selon Gasser [Gasser, 1990], une société d'agents est constituée de trois éléments : un ensemble d'agents, un ensemble de taches a réaliser et un ensemble d'objets associés à l'environnement. Un agent peut prendre la responsabilité d'effectuer une tâche s'il en a la capacité. Il prend alors un

rôle dans le groupe. La réalisation d'une tâche suppose la manipulation d'objets de l'environnement.

3.2 La Coopération

De par le fait que les SMA sont conçus dans le but de résoudre un problème de manière distribuée, il apparaît que la coopération est l'une des caractéristiques essentielles dans un SMA. Aussi elle est la forme d'interaction la plus étudiée.

Dans un SMA, chaque agent renferme une expertise qui ne lui permet pas en général de résoudre seul le problème posé. Il ne peut que participer partiellement à la résolution, et donc soit aider un autre agent à résoudre un sous problème, soit de demander à un autre de l'aider à résoudre un sous problème. Ceci introduit la notion de coopération entre les agents et pose la question : qui fait quoi, quand, par quel moyens et sous quelles conditions ?

3.2.1 Définitions

La coopération est un accord entre agents permettant la synchronisation des activités des agents, le parallélisme des calculs et le partage des ressources.
[ScaBar96]

La coopération dans un SMA est développée par la volonté des agents de participer à l'accomplissement d'un objectif commun. Elle se rapporte à l'activité globale d'un groupe d'agents car l'aboutissement de la résolution distribuée d'un problème est le résultat de l'interaction coopérative entre les différents agents. [Della,2000]

Des agents coopèrent ou sont en situation de coopération si les conditions suivantes sont vérifiées : [Labidi et Lejouad, 1993]

- l'ajout d'un nouvel agent permet d'accroître les performances du groupe ;
- l'action des agents sert à éviter d'éventuels conflits.

Ainsi, un agent dispose de moyens qui lui permettent de s'informer sur l'état de l'environnement et des moyens de modifier l'état de cet environnement.

Les objectifs de la coopération entre agents est d'améliorer le mode d'évolution de la résolution (par les agents) en termes : [Baujard,1992]

- de validité et de rationalité des informations échangées et des comportements ;
- d'efficacité des stratégies de résolution employées (alternance entre comportement opportuniste et comportement dirigé par les buts) ;
- de suppression des efforts inutiles de synchronisation ;
- de cohésion entre planification locale et globale.
- de rééquilibrage dynamique de la charge du travail (attribution de tâches selon disponibilité) ;

Un agent doit disposer – en plus de la connaissance reflétant son degré d'implication dans cette dynamique collective (croyances, buts, intentions, engagements, modèle de soi et d'autrui) - d'un certain nombre de compétences nécessaires pour la coopération. Il doit pouvoir [Labidi et Lejouad, 1993]:

- mettre à jour le modèle du monde environnant,
- intégrer des informations venant d'autres agents,

- interrompre un plan pour aider d'autres agents,
- déléguer la tâche qu'il ne sait pas résoudre à un autre agent dont il connaît les compétences.

Ces caractéristiques forment les qualités essentielles d'un agent coopératif.

3.2.2 Les modèles de coopération

Selon qu'il existe une hiérarchie ou non entre les agents, nous distinguons les modes de coopération suivantes :

3.2.2.1 Coopération en hiérarchie

3.2.2.1.1 Le mode commande

L'agent superviseur décompose le problème en sous problèmes qu'il envoie aux agents exécutants. Ces derniers, après avoir résolu leur problème, renvoient le résultat au superviseur qui choisira la «meilleure » réponse.

3.2.2.1.2 Le mode compétition

L'agent superviseur décompose le problème en sous problèmes et envoie la liste des sous problèmes à tous les agents exécutants. Ceux-ci consultent la liste et en résolvent chacun, un ou plusieurs. A la fin, ils renvoient leurs solutions respectives au superviseur qui décident des solutions adoptées.

3.2.2.1.3 Le mode appel d'offres

La répartition de sous problèmes se fait comme dans les autres modes. Chaque agent envoie pour sa part un appel d'offre après avoir consulté la liste. L'agent superviseur, à son tour, envoie les sous problèmes aux agents qu'il aura chois C'est seulement, en ce moment que les agents exécutants résolvent leurs sous problèmes et envoient leurs résultats au superviseur.

3.2.2.2 Coopérations sans hiérarchie

3.2.2.2.1 Coopération par partage de tâches

Le contrôle est dirigé, ici, par les buts. Le problème est réparti parmi tous les agents qui travaillent parallèlement, disposant des compétences et des ressources nécessaires. Chaque agent représente une tâche qu'il s'est engagé à exécuter. Donc à chaque engagement, les buts de l'agent changent et puisque les agents sont autonomes, on dit que le contrôle est dirigé par les buts.

3.2.2.2.2 Coopération par partage de résultats

Dans ce mode de coopération, le contrôle est dirigé par les données. Aucun agent ne peut résoudre une tâche individuellement. Il nécessite donc l'expertise des autres, donc ils s'échangent des solutions partielles à la demande. Chaque agent est représenté par une source de connaissance.

3.3 Le Contrôle

Le mécanisme de contrôle dans un SMA doit permettre de résoudre des problèmes de plusieurs types [Vincent, 1993] :

- après avoir résolu un problème posé, le résultat peut influencer la décision de l'autre ;
- les buts d'un agent et ceux du système sont antagonistes ;
- plusieurs agents ont besoin d'une ressource qui n'est pas partagée ou qui est limitée ;
- on peut arriver à une situation de conflit puisque dans certains systèmes les solutions des agents sont exclusives ;

Dans les systèmes où le contrôle est centralisé, des problèmes importants qui pourraient se poser sont la résistance aux pannes de tout le système. Si l'agent contrôleur s'effondre, tout le système s'écroule.

Dans les systèmes où le contrôle est réparti entre plusieurs agents des problèmes de coordination des actions peuvent surgir.

Un mécanisme de contrôle efficace doit pouvoir choisir l'agent le plus apte à une étape donnée de la résolution. Par conséquent, il est rapide (en terme d'action), économique (gestion des ressources et nombre de communications effectuées) et il doit satisfaire le plus de contraintes dans un problème où la solution est sur contrainte.

Selon Vincent [Vincent, 1993], le contrôle ou encore conduite du raisonnement représente les activités d'organisation du système et de répartition des rôles et tâches parmi les agents afin que le système aboutisse à une solution. Il concerne la planification et le suivi des tâches lors de la résolution.

Le contrôle peut être conduit de différentes manières. Il peut être :

- par un seul (moniteur) qui gère l'ensemble des agents. Il doit être capable de choisir à tout instant lors de la résolution l'agent qui sera chargé de faire évoluer vers la solution ;
- par un groupe d'agents appelés «agents de contrôle » dont le rôle de chacun consiste en la gestion d'un sous-groupe d'agents ;
- par tous les agents parmi lesquels les connaissances de contrôle sont distribués et par conséquent la négociation est accentuée pour la résolution .

Suivant le nombre d'agents ayant une activité de contrôle sur le domaine, le contrôle sera centralisé ou décentralisé

3.3.1 Contrôle centralisé

Un seul agent se charge du contrôle donc décide des actions à entreprendre, gère les conflits et les ressources. Tous les autres agents sont sous sa commande et n'entament une tâche que s'il le leur demande. L'autonomie de l'agent «résolveur » ne se fait valoir qu'après l'allocation de la tâche. En outre, l'agent « moniteur » se réserve le droit d'interrompre la tâche d'un agent (en ne prenant plus en considération la solution rendue par celui-ci).

Ce processus simple permet de conduire et de prédire les interactions des différentes entités du système autour du «moniteur ». Le contrôle est alors dit centralisé et implique un système très efficace s'il n'est pas surchargé. Il est aussi assez facile à mettre en œuvre.

Ce type de contrôle pose des problèmes dans la communication et le traitement, car cet agent constituera un goulot d'étranglement. Sa fiabilité est aussi compromise car s'il est en panne, il entraîne tout le système. On trouve ce modèle dans les systèmes à tableau noir.

3.3.2 Distribution du contrôle

La question «quel agent fait quoi et quand ?» peut être considérée comme étant la principale question en IAD. Ainsi, la distribution du contrôle se fait selon trois caractéristiques [Vincent, 1993] : le degré de coopération entre les agents, le type d'organisation adapté et la manière de la mise en œuvre de cette coopération.

Premièrement, la distribution des activités du contrôle peut entraîner que les agents aient des buts locaux antagonistes entre eux ou avec le système. Un agent est coopératif s'il est capable de changer ses objectifs locaux pour satisfaire ceux d'un autre agent ou ceux du système. Mais cette distribution pourrait ne pas être bénéfique puisque la coopération peut être très coûteuse en terme de communication.

Deuxièmement, le type d'organisation utilisé est étroitement lié au contrôle. Les relations et la répartition des rôles par le contrôle définissent les types d'organisation. Dans une organisation hiérarchisée, les agents renfermant plus d'information guident (contrôlent) ceux qui en ont moins. Les relations d'autorité du supérieur au subordonné surgissent entre agents [Vincent, 1993]. Elles seraient d'autant plus importantes si l'autorité revient aux agents qui ont une vue globale du système. Un problème appelé principe de rationalité limité ou d'éventail de contrôle se passe alors. Cela veut dire qu'un agent ne peut avoir d'autorité que sur un nombre limité d'autres agents. Dans les entreprises humaines, on l'appelle éventail de subordination ou surface de contrôle. On trouve ce modèle dans les systèmes d'acteurs.

3.4 La Communication

La communication est la base de la résolution coopérative des problèmes. Elle permet de synchroniser les actions des agents et résoudre les conflits de ressources et de buts par la négociation. Dans les systèmes multi-agents, les agents ne disposent d'aucune mémoire commune. La communication entre les agents repose explicitement sur des mécanismes d'envoi de message, de réception et de synchronisation .

3.4.1 Protocoles de communication

Dans un système distribue, il faut que les entités disposent d'un langage commun pour pouvoir échanger des informations et coopérer pour la résolution d'un problème. Ce langage appelé mécanisme de communication interprocessus est un ensemble de primitives connues par chaque entité et dont l'ensemble constitue un protocole de communication.

Les primitives de base d'un protocole de communication sont les suivantes [Labidi et Lejouad, 1993]:

- établissement d'une connexion entre deux entités ;
- identification du nœud destinataire dans un réseau de communication ;

- envoi de données ;
- réception de données ;
- définition d'un type de communication : synchrone ou asynchrone.

Les protocoles de communication sont les règles nécessaires à la communication, ils permettent de structurer et d'uniformiser les interactions inter-agents. Un protocole de communication peut être basé sur une architecture standard de communication, en l'occurrence, Internet ou Os

3.4.2 Modes de communication

Il existe deux principaux modes de communication : la communication par envoi de messages et la communication par partage d'informations.

3.4.2.1 Communication par envoi de messages

Les agents sont en liaison directe et envoient leurs messages directement et explicitement au destinataire. La seule contrainte est la connaissance de l'agent destinataire : « Si un agent A connaît l'agent B alors il peut entrer en communication avec lui » [Labidi et Lejouad, 1993].

Les systèmes fondés sur la communication par envoi de messages relèvent d'une distribution totale à la fois de la connaissance, des résultats et des méthodes utilisées pour la résolution du problème. Les systèmes d'acteurs en sont l'illustration parfaite.

Ce type de communication concerne aussi bien les agents cognitifs que les agents réactifs. Les agents échangent des informations en envoyant et en recevant des messages. Ces échanges peuvent se faire explicitement entre deux agents (point à point), d'un agent à un groupe d'agents (Multicast) ou d'un agent à tous les autres agents (broadcast). Pour cela, les agents disposent d'un même protocole de communication. Ils ont, en outre, une base de connaissance locale sur leurs accointances (les autres agents). Donc un agent peut à tout moment, s'il le juge nécessaire faire des envois de message qui peut être une information ou une requête. Nous nous trouvons dans un cas où il y a une distribution des connaissances.

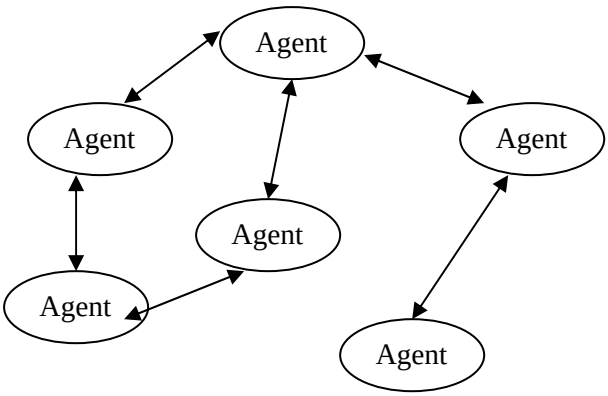


Fig. 6 : Communication par envoi de messages

3.4.2.2 Communication par partage d'informations

Ce type de système de communication concerne les agents les plus évolués notamment les agents cognitifs. Il est mis en évidence par le fait que les agents communiquent à travers une structure de données contenant les données initiales du problème et au fur et à mesure qu'on avance vers la solution, on y stocke les solutions partielles des différents agents. Cette structure est souvent appelée Blackboard tous les agents ont accès à la structure de donnée ; ils y prennent les informations qui les intéressent, les traitent puis déposent leur solution dans la structure. De cette manière, les agents ne sont pas au courant de l'existence de leurs paires.

Les agents sont anonymes en ce sens qu'il n'est pas nécessaire à un agent de savoir qui a émis telle ou telle information. Les agents ne sont en relation directe qu'avec le tableau noir dont ils sont connectés par le système de contrôle. Par conséquent, il est plus facile de les supprimer ou d'en ajouter dans le système car seul le BB doit en être informé.

Un problème qui doit être relevé est le fait qu'un libre accès au BB amène les agents à traiter un grand nombre de solution partielle. Donc la dynamique de résolution est organisée en niveau.

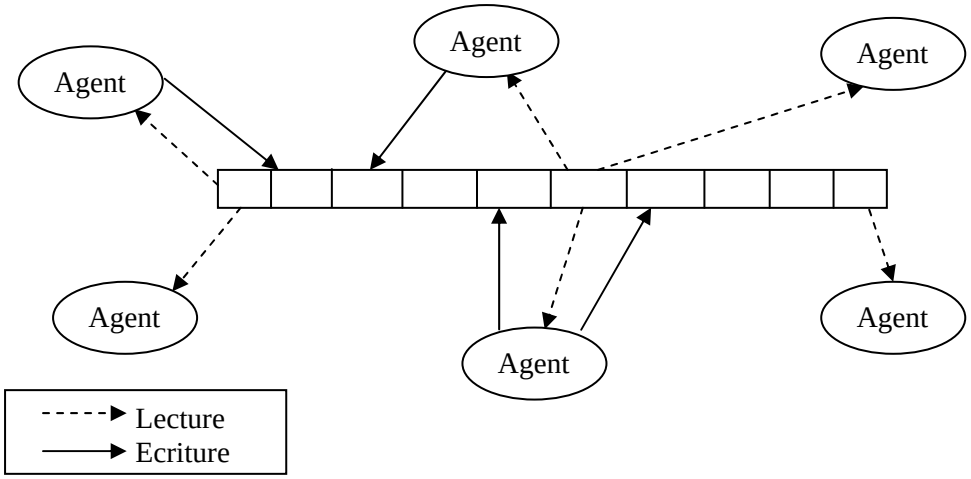


Fig. 7 : Communication par partage d'information

3.4.2.3 Délégation

Les accointances sont spécifiques à chaque agent. A la réception d'un message, l'agent vérifie si l'expéditeur du message fait partie de ses accointances. Si l'expéditeur est inconnu, l'agent ne va pas traiter le message, mais le déléguer à un autre susceptible de s'y intéresser. Pour cela, il consulte les informations qu'il possède sur les modèles des autres agents.

La délégation est un mécanisme asynchrone et non bloquant, permettant à chaque acteur, ne pouvant pas traiter un message, de le déléguer à un autre acteur, appelé PROXY

[Labidi et Lejouad, 1993]. Le PROXY prend en charge la tâche qui lui a été déléguée et l'acteur déléguant est donc immédiatement prêt à traiter un autre message. Le PROXY connaît toutefois l'acteur déléguant, pour l'interroger dans le cas où il aurait besoin d'information supplémentaire.

La délégation est un moyen de partage de connaissances et de comportement plus souple que la notion d'héritage. Un acteur peut se choisir dynamiquement de nouveaux PROXYs, alors que le partage de connaissances établi par le graphe d'héritage est statique.

3.5 La résolution de conflits

Les agents coopératifs ont besoin d'éviter autant que possible les situations conflictuelles pour résoudre un problème, Chaque agent a des buts et des objectifs qui lui sont propres et ils peuvent être en conflit avec ceux des autres. Dans le souci de résoudre le problème, une résolution de conflit est nécessaire. Ils concernent la négociation et coordination d'actions. La première s'occupe de la manière dont les agents s'échangent des informations et la seconde traite de la façon dont les actions des différents agents doit être organisées dans le temps et dans l'espace de manière à réaliser les objectifs [Ferber, 1995].

3.5.1 La négociation

La négociation correspond au dialogue entre agents afin de concilier des vues incohérentes et d'obtenir un compromis sur la manière dont les agents doivent agir ensemble. [Della,2000]

K.Sycara [Sycara, 1989] la définit comme étant un processus itératif qui implique l'identification des itérations potentielles par communication ou par raisonnement sur les états constants et intentions des autres agents du système et la modification de ces intentions de manière à éviter les interactions nuisibles et créer les situations coopérantes.

La négociation a été introduite dans le but de conjuguer les intérêts individuels des agents souvent conflictuels de manière à satisfaire les contraintes du problème en désignant les sources du conflit de telle sorte qu'on y pallie sans pour autant modifier les buts des agents.

La négociation est un processus adaptatif (instant du déclenchement du conflit inconnu), sûr car l'indisponibilité d'un agent ne bloque pas le processus et efficace car les communications sont structurées de telle sorte qu'elles minimisent les échanges [Vincent, 1993]. La communication et l'organisation influencent beaucoup le type de négociation en fonction des politiques. La négociation permet aux agents d'avoir un contrôle distribué sur les données incohérentes (du domaine) et d'avoir un contrôle sur les stratégies (changement) de résolution (allocation de tâches). La négociation est caractérisée par: [Labidi et Lejouad, 1993]

- Un faible nombre d'agents impliqués dans le processus, la plupart des travaux ne mettant en œuvre que deux agents négociants ;
- La négociation implique au moins le protocole d'actions suivantes : proposer, évaluer, corriger et accepter une solution ;

- Le processus peut se dérouler de manière distribuée ou centralisée. Dans ce dernier cas, un agent est impliqué en tant qu'agent négociateur, il est toujours le même ou désigné en fonction du contexte.
- Afin de permettre les échanges d'information, un langage commun est nécessaire ;
- Les agents peuvent avoir besoin du modèle des autres agents pour négocier ;
- La négociation ne consiste pas forcément à trouver un compromis à partir des positions initiales mais peut également consister à modifier les croyances de l'autre agent pour faire prévaloir son point de vue.

3.5.2 La coordination

La résolution d'un problème par un ensemble d'agents doit être temporairement coordonnées pour qu'elle soit utilisée par d'autres agents.

La coordination est définie comme étant l'ensemble des activités supplémentaires qu'il est nécessaire d'accomplir dans un environnement comprenant plusieurs agents et qu'un agent seul poursuivant les mêmes buts n'accomplirait pas. [Vincent, 1993].

La coordination est l'activité qui permet aux agents de considérer toutes les tâches (aucune tâche n'est ignorée) et de ne pas dupliquer le travail [Labidi et Lejouad, 1993].

La coordination est construite par les agents en fonction de leurs différents objectifs et en conciliant les contraintes locales des agents tout en permettant la résolution d'un problème global.

Les systèmes d'allocation de tâche et de coordination tels que la médiation (médiateurs ou courtiers reliant les agents) et le réseau contractuel basé sur le marché récupèrent les résultats d'un agent et cherche celui à qui il doit affecter la tâche suivante. Il y a là une certaine planification (distribuée).

Les agents coopèrent avec une négociation et une coordination d'action qui leur permet d'éviter au maximum les conflits. C'est ce principe qui est adopté dans le modèle du Partial Global Planning (PGP). Les agents s'échangent leurs plans partiels respectifs permettant à chacun de modéliser les activités des autres.

3.6 L'Emergence

L'émergence est un concept relatif aux systèmes réactifs. C'est un phénomène complexe et très peu compris, il est en cours d'exploration dans les systèmes multi-agents réactifs ; il est aussi étudié par beaucoup de chercheurs en physique, en biologie et en systématique.

Une fonctionnalité qui émerge est une fonctionnalité qui n'existait pas au départ et qui n'a pas été explicitement programmé, elle est le résultat d'interaction entre deux ou plusieurs comportements [Ferber, 1995]. Prenons cet exemple introduit dans [Labidi et Lejouad, 1993] : si on veut qu'un robot (en marche) suive un mur (comportement suivre le mur), on définit deux comportements (antagonistes) : être attiré par le mur et être repoussé par le mur. De l'interaction de ces deux comportements va émerger un nouveau comportement : suivre le mur.

Dans ce cas, on parle d'émergence de comportement, mais il existe aussi d'autres formes d'émergence telles que l'émergence de rôle, l'émergence de stratégie, l'émergence de contrôle et émergence de structures de coordination.

4 Conclusion

Nous avons vu dans ce chapitre, les différents concepts qui entrent en jeu lors de la mise en œuvre d'un Système Multi Agents. En fait, nous avons remarqué que ce sont des concepts souvent interdépendants. En effet, la communication est à la base de toutes les transactions entre agents. Or, la résolution distribuée de problème nécessite une certaine organisation et une coopération entre agents, c'est-à-dire l'échange d'information relative aux solutions partielles. Le contrôle doit mener le processus de résolution et ainsi, assurer la coordination et une négociation pour éviter ou résoudre les conflits occasionnés par les buts différents agents.

Nous avons aussi abordé le sujet de l'émergence d'intelligence dans les systèmes à agents réactifs. C'est un concept qui n'est pas très maîtrisé dans la pratique. Il se trouve toujours au stade de recherche. Pour le problème de la poursuite de Cliff & Miller, on va faire une extension multi-agents qui se base sur l'ensemble des concepts et techniques introduites dans ce chapitre, et plus exactement, la communication entre les agents surtout dans le cas où les agents peuvent être de différents types.

La proposition de solution, méthodes et techniques
L'approche COCAGE

(*Collective Co-evolution based on Multi-Agents Environnement Approachs*)

1. Introduction

Après avoir exposé le problème de poursuite, sa puissance de modélisation, ses limitations ainsi que ses extensions, on voit intéressant de le généraliser dans un contexte pluri entités (plusieurs Poursuiveurs et plusieurs Poursuivis), qui dit pluri entités dit multi-agents, mais un inconvénient majeur des agents dans les systèmes multi-agents est la non-existence de morphologie intrinsèque d'agent. En contrepartie l'avantage des systèmes multi-agents est les concepts puissant de communication et d'autres concepts déjà cités dans le chapitre sur les systèmes multi-agents .

La définition d'une morphologie d'agent donne un vrai sens à la propriété d'autonomie bien connue dans la définition d'agent, une morphologie pour l'agent donne un nouveau type de systèmes multi-agents, un SMA hybride, des agents évolutifs dans système communiquant.

Notre proposition pour modéliser la généralisation du problème de poursuite de Cliff & Miller, est faite sur deux volets, le premier volet qui est principal c'est la définition d'une architecture de l'agent la deuxième est une architecture du système d'agents. Ces deux extensions vont données une infrastructure de poursuite « COCAGE ». On va donc bénéficier de la puissance des systèmes multi-agents et des qualités des systèmes évolutifs pour trouver une solution aux critiques et limitations de la poursuite de Cliff&Miller.

2. La proposition d'architecture d'agent

L'approche cognitiviste et l'approche comportementale sont habituellement présentées comme les deux courants opposés dans le domaine du contrôle de systèmes autonomes ou multi-agents en IAD.

D'autres travaux tendent à intégrer les deux courants en une troisième approche dite hybride, cette dernière souffre du manque de morphologie signalée ci-dessus.
Maintenant on va exposer les approches utilisées dans notre architecture d'agent, commençant du plus haut niveau.

2.1 L'approche cognitiviste

Notre agent va avoir un niveau cognitif symbolisé par un raisonnement d'expert.

L'approche Cognitive est Basée sur la métaphore calculatoire, qui suppose un niveau d'abstraction dans lequel les processus mentaux peuvent être considérés indépendamment des mécanismes qui les engendrent, ce type d'architecture de contrôle de systèmes autonomes est apparu avec l'avènement de l'Intelligence Artificielle (1956).

Le traitement de l'information y est décomposé en trois *fonctions* successives:
La perception, la planification et l'exécution. L'architecture exploite un *modèle du mond*e, exprimé en général dans un formalisme logique, donné par le concepteur et mis à jour par la perception. Cette connaissance correspond au point de vue du concepteur sur un certain domaine d'activité et représente souvent les limites de l'expertise humaine.
Elle décrit des objets du monde, leur état et leurs propriétés, des éléments de savoir-faire, ou des méta connaissances.

Dans ce cas, la représentation est dite objectiviste dans le sens où les informations entrant dans le système sont à propos du monde tel qu'il est donné par le concepteur.

La mise à jour de cette connaissance objectiviste, la complexité des algorithmes de planification (exponentielle) et la lourdeur des mécanismes de récupération (re-planification) produisent des systèmes lents et peu flexibles face à la dynamique et aux imprévus du monde réel.

Ceci explique la simplicité des mondes dans lesquels ces systèmes sont testés et validés.
De nombreux travaux ont tenté de réduire ces déficiences soit par une prise en compte du temps (planification réactive), soit en raisonnant non seulement sur le problème donné mais également sur les techniques disponibles pour engendrer un plan (méta planification).

2.2 L'approche comportementale

Notre agent va avoir un niveau comportementale symbolisé par un module de condition-action.

Face aux limitations de l'approche cognitiviste, une partie de la communauté IA s'est intéressée, dès 1980, à développer des systèmes pouvant agir rapidement dans des univers dynamiques et imprévisibles.

L'approche comportementale propose des architectures décomposées en modules (comportements) réalisés sous la forme de règles (condition-action) ou de boucles sensori-motrices (liant les capteurs aux effecteurs par un traitement simple).

Les boucles sensori-motrices sont concurrentes et pose le problème de l'arbitrage entre celles-ci.

Trois types génériques de contrôle peuvent être utilisés:

Contrôle par priorité

Les comportements concurrents agissent sur le monde en fonction de certaines priorités préétablies. Deux exemples existent :

- Un tel mécanisme avec des règles contrôlant le comportement d'un pingouin (la proie) dans une grille peuplée de cubes de glace (obstacles ou armes) et d'abeilles hostiles (les prédateurs).
- La *subsumption architecture* introduit des priorités précablées (circuits inhibiteurs ou suppresseurs) entre boucles hiérarchisées en niveaux de compétences (évitement d'obstacles, sensori-motrices déambulation, exploration, …). [Brooks91]

Contrôle par compétition

Les comportements concurrents sont interconnectés (en réseaux) et prennent le dessus en fonction d'une dynamique interne au système. On peut Citer de exemples :

[Maes89], par exemple, propose un réseau d'opérateurs se propageant différentes activations (+/-) par des liens 'successeur', 'prédécesseur' et/ ou 'conflictuel', suivant les interactions établies par la donnée des pré-conditions et de chaque opérateur.

Ces activations proviennent aussi bien de post-conditions la situation observée que des buts courants.

[Beer90] propose une architecture neuromimétique faite de neurones et de connexions synaptiques par lesquelles un neurone peut influencer la connexion entre deux autres neurones soit en l'interrompant, soit en modifiant la force de la connexion.

Contrôle par fusion

Les consignes proposées par l'ensemble des comportements sont fusionnées en une consigne commune, compromis entre les différentes tendances comportementales.
Ce type de contrôle nécessite une représentation uniforme permettant d'exprimer les tendances et de calculer leur fusion.
Chez [Anderson90], les consignes peuvent être représentées vectoriellement (direction et force) et la fusion se résume alors à une sommation vectorielle pondérée.
[Payton90] propose une fusion des tendances de l'agent par propagation d'un gradient depuis chaque objectif localisé sur une carte.
Les différents gradients sont fusionnés par sommation vectorielle en tout lieu de la carte.
Ces trois types génériques de contrôle sont parfois combinés à travers une même architecture : [Tyrrell93], par exemple, combine les contrôles par priorité et par fusion en une architecture hiérarchique.

2.3 L'approche Evolutionniste

Notre agent va avoir un niveau Evolutionniste symbolisé par un module générateur de morphologie et de contrôle.

Bien que les comportements soient généralement pré-programmés et figés dans les systèmes qui existent, on ne peut pas bénéficier de la puissance de l'évolution et de la co-évolution cités dans le chapitre « la poursuite de Cliff & Miller ». On a adopté une approche évolutionniste et on s'intéresse à l'acquisition par le système lui-même de boucles sensori-motrices et la morphologie complète de l'agent, position des capteurs, position des roues le contrôleur neuronale et le codage génétique des contrôleurs …, ce qui va mettre notre système dans un contrôle adaptatif et en évolution permanente.

2.4 L'approche hybride

Certains auteurs ont intégré l'approche cognitiviste et l'approche comportementale en une architecture à deux niveaux: un niveau réactif (comportemental) surmonté d'un niveau de représentation (cognitiviste) pour l'arbitrage.
L'objectif est d'associer la réactivité des systèmes comportementaux à la capacité de raisonnement et d'organisation des systèmes cognitivistes.
Les systèmes hybrides se différencient essentiellement par le type de représentation qu'ils utilisent:

2.4.1 Architecture de l'agent

L'architecture générale que nous avons développée, est décomposée conceptuellement de quatre niveaux d'abstraction: le niveau générateur de l'Agent (0), le niveau physique (1), le niveau comportemental (2) et le niveau dit cognitif (3).

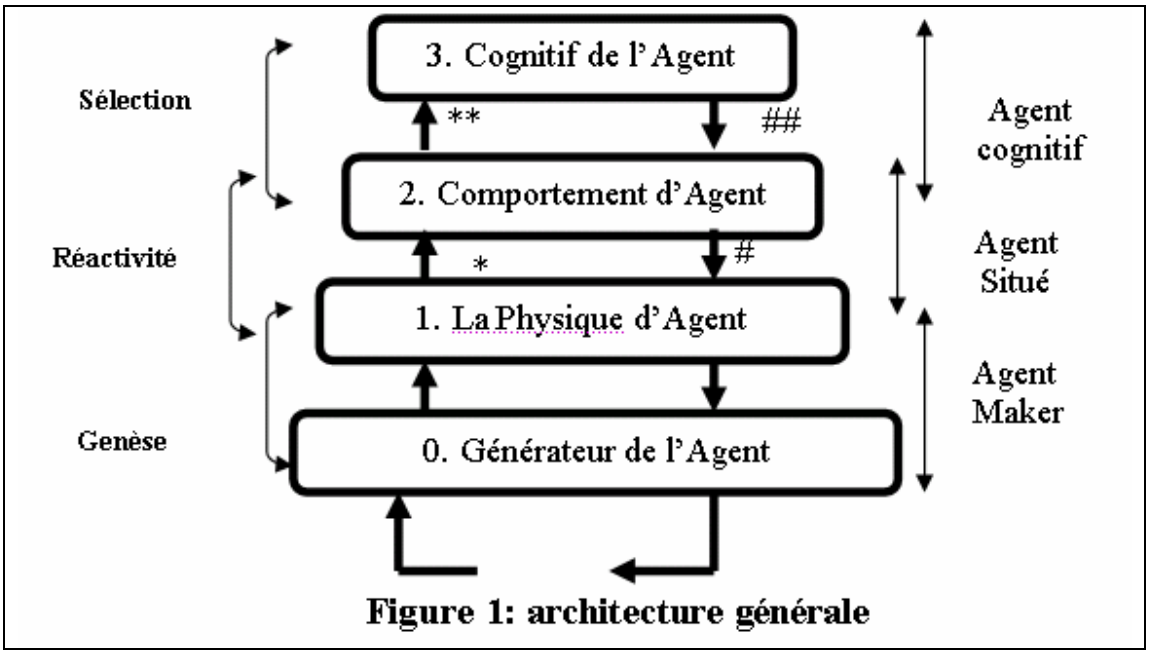


Figure 1: architecture générale

En ce sens, il s'agit d'une architecture hybride (voir figure 1).

Le *niveau 0* définit l'agent Maker composé d'un module évolutionniste pour la création d'un contrôleur neuronal dédié à contrôler une morphologie elle-même créer par ce niveau. Ce dernier est un module de bas niveau qui donne une autonomie complète à notre architecture d'agent.

Le *niveau 1* définit l'agent physique, composé de ses capteurs et de ses effecteurs. C'est lui qui établit le contact avec le milieu.

Le *niveau 2* contient les boucles de l'agent. Il introduit la dimension de réactivité face à la dynamique sensori-motrice de l'environnement et fait de l'agent physique, un agent situé, c'est-à-dire en interaction constante avec son milieu.

Enfin, le *niveau 3* est composé de la connaissance et de trois processus cognitifs (l'interprétation, l'apprentissage et la motivation) qui structurent cette connaissance. Il introduit la capacité de sélection et fait de l'agent un agent cognitif (mais toujours sans référence à un monde extérieur), exhibant certaines dimensions d'autonomie.

Chaque niveau d'abstraction fournit des informations au et est contrôlé par le niveau immédiatement supérieur.

2.4.2 Les communications entre les couches

L'agent Maker crée une morphologie et un contrôleur suivant les résultats obtenus dans la génération précédente. Si c'est la première génération l'agent maker commence par une solution initiale comme toute approche génétique.
Ainsi, la liaison * transmet l'état des capteurs (signaux) aux boucles sensori-motrices. Ces derniers traitent les signaux reçus et génèrent des commandes qui, envoyées par la liaison #, permettent aux boucles sensori-motrices de contrôler directement les effecteurs.
De même, la liaison ** transmet l'état des boucles sensori-motrices (caractéristiques perçues et état de stimulation) au niveau cognitif qui, par la liaison ##, sélectionne la boucle sensori-motrice qui lui semble adéquate.
A tout instant, seule la boucle sensori-motrice sélectionnée par le niveau cognitif contrôle les effecteurs du niveau physique.
Cette architecture s'éloigne fondamentalement des architectures de subsomption dans la ligne des travaux de Brooks dans lesquelles les boucles s'inhibent les unes les autres.
En effet, les boucles sensori-motrices n'interagissent jamais l'une avec l'autre et la couche supérieure ajoute un niveau « symbolique » sans altérer les fonctionnalités préexistantes.
Ceci est compatible avec la compréhension contemporaine de l'évolution selon laquelle les changements qualitatifs se superposent sans les altérer aux fonctions sous-jacentes.

Cette architecture présente des propriétés intéressantes:

a. Approche ascendante ou synthétique

Elle a été conçue de bas en haut. Les modules de génération de la physique sont la base des capacités physiques sont la base (contraignante) des comportements qui sont, à leur tour, la seule base (contraignante) de la cognition.
Ces dépendances imposent des choix méthodologiques déterminants quant aux représentations envisageables au niveau cognitif.

b. Modularité conceptuelle

La réactivité et la sélection y sont distinctement séparées à travers les niveaux d'abstraction. Ceci facilite la réflexion quant aux besoins de chaque niveau et à leurs interactions.

c. Architecture hybride

Elle intègre, de façon naturelle, l'approche comportementale et l'approche cognitiviste de la modélisation d'agents autonomes en morphologie et en contrôle.

d. Généralité

Le contenu de chacun des quatre niveaux d’abstraction reste ouvert. L’architecture est indépendante de l’entité qu’il utilise, des boucles sensori-motrices et du modèle cognitif utilisés.

Seules sont spécifiées les interfaces entre les niveaux d'abstraction.
Plateforme d’étude pour le problème de la « sélection d’action ».

Notre architecture se prête particulièrement bien à l’étude générale de ce sujet: l’interface de contrôle y est clairement spécifiée et le niveau cognitif n’impose aucun modèle cognitif particulier.

Cette architecture peut donc servir à l’étude et à la comparaison de différents modèles cognitifs sur la base d’une même plateforme comportementale.

e. La représentation

Classiquement, une représentation est un modèle de certains aspects d'une réalité extérieure. Elle doit obéir au moins à deux critères:

Le critère de *pertinenc*e:

On ne s'intéresse qu'aux aspects qui sont suffisants pour répondre à une certaine question ou réaliser une certaine tâche.

Le critère *d'adéquation* ou de réalisme

Les aspects que l'on modélise doivent être conformes à la réalité. Le deuxième critère suppose un observateur capable de voir à la fois la réalité et le modèle ce qui, d'une certaine façon, est le cas lorsqu'un concepteur réalise un système artificiel. C'est le point de vue *objectivist*e. Si nous voulons que notre système soit autonome, il doit élaborer ses propres représentations en étant incapable d'accéder par lui- même à une quelconque « réalité ».

Seule l'expérience de son interaction avec l'environnement à travers ses capteurs et ses effecteurs peut servir de base à une élaboration des représentations, sans référence a priori avec un quelconque monde extérieur.

1. Dans les générateurs de morphologie et de contrôle on s’intéresse à créer de agents physiques qui dépassent leurs échecs dans la génération précédente.

2. Dans les boucles sensori-motrices qui sont des réponses à des stimuli de façon à maintenir un invariant sensoriel, et qui constituent les éléments sensoriels de base.

3. Dans le niveau cognitif par la structuration topologique de la succession des boucles sensori- motrices applicables.

Succinctement, une boucle sensori-motrice *b* peut être applicable ou non selon la présence du stimulus correspondant.
Un état instantané *e* est l'ensemble des états d'applicabilité de chacune des boucles :

$$e = \{applicabilité\ (\ bi)\}.$$

Un historique *h* est une succession d'états instantanés et de choix d'une des boucles sensori-motrices applicables, une boucle restant sélectionnée tant qu'elle reste applicable :

$$h=(\ e1, b1, e2,\ldots\ en, bn).$$

On dira alors que la sélection d'une boucle applicable *bi* dans l'état *ei* a causé le nouvel état *e(i+ 1*).
Si ce n'est pas le cas, un morceau de l'historique représentera l'état instantané *x* dans lequel le choix d'une boucle produit la transition souhaitée.
Par exemple, si on est passé de *e(* à *e(i)* en sélectionnant *b(i- 1)* alors on arrive en *e(i+ 1)* en appliquant *bi* (dans ce cas *i-1) xi = (e(i- 1), b(i- 1), ei)*).
Le critère de *pertinence* est respecté puisque les stimuli ne sont pas arbitraires mais pertinents à la réalisation de comportements (suivi de mur, évitement d'obstacle, attraper les poursuivis, échapper aux poursuiveurs …) et que la structuration des successions est pertinente au problème du choix de la prochaine boucle sensori-motrice à sélectionner.
Le critère *d'adéquation* doit être adapté pour ne pas faire référence à une réalité inaccessible: nous dirons qu'une représentation est adéquate si elle rend compte de l'histoire de l'interaction du système avec son environnement (c'est-à-dire est compatible avec h).
La représentation utilisée au niveau cognitif et gérée par les processus cognitifs possède quatre caractéristiques essentielles:

a) Elle est extraite des informations provenant du niveau sensori-moteur et est, par conséquent, de nature sensori-motrice,
b) Elle porte sur la perception par l'agent de son interaction avec son milieu (interaction monde-agent) par opposition à la perception directe du milieu,
c) Elle exprime les régularités causales de cette perception,
d) Elle sert à prédire les événements sensoriels futurs.

Ces caractéristiques en font un excellent exemple de connaissance subjectiviste (par opposition aux modèles objectivistes généralement usités en IA).

Les trois processus cognitifs que nous avons considérés -interprétation, apprentissage et motivation- s'occupent respectivement :

i. D'analyser les informations sensorielles provenant des boucles sensori-motrices,
ii. de structurer ces informations en un graphe causal représentant la topologie de l'interaction,
iii. De situer les objectifs courants sur cette représentation et d'orienter les choix de l'agent.

Après avoir défini l'architecture d'agent pour la poursuite, la définition des systèmes multi-agent devient simple, il suffit de modéliser la communication et la nature de chaque entité des deux classes d'agents (agent poursuiveur et agent poursuivi).

2.5 Modélisation de la communication d'agent (voir chapitre VI §3.4)

La communication entre agents crées par le processus de morphogénéisation, est modélisé comme suit, dans le génotype des agents communiquant (niveau générateur de l'agent) on ajoute un champ qui caractérise l'évolution des capteurs pour la communication, c'est le même cas pour les capteurs de distance.

Chaque capteur de communication a comme attributs la portée et champs qui contiennent l'information communiquée ou à communiquer. La portée des capteurs est exprimée par un champ d'attraction dans lequel l'agent peut recevoir et émettre de l'information à tous ses voisins de même type. Pour les agents non communiquants, ce champ est simplement inactif, pour qu'il n'intervienne pas dans la morphogénéisation (le passage du génotype au phénotype).

Les informations transmises entre les poursuiveurs sont la position ,et l'orientation relative ou absolue de chaqun des poursuivis qui sont dans le champ de vision, à noter que généralement le champ de vision est plus étendu que le champ de communication (portée).

Dans les systèmes Multi-agents, même s'il n'y a pas de communication directe, en peut toujours envisager des communications indirectes, qui elle sont pas propres aux agents, comme exemple, une communication via le monde, par récompense, par apprentissage et d'autre techniques.

3. Proposition de la solution Multi-agents

Notre solution est plus une modélisation complète pour tous les aspects d'une poursuite dans le cas général, « Un Système de Poursuite », c'est-à-dire, plusieurs poursuiveurs avec plusieurs poursuivis dans un monde infini à deux dimensions ou un monde à base de cases et de déplacement possibles. Pour rester dans un contexte générique on va voir que la solution Multi-agents est faite sur la base de deux concepts importants :

- Homogénéité des agents de chaque type (Poursuiveurs, Poursuivis),
- La communication entre les agents (voir chapitre VI §3.4).

Voici un tableau explicatif :

	Communiquants	Non Communiquants
Homogènes	*SMA homogène communiquant*	*SMA homogène non communiquant*
Hétérogènes	*SMA hétérogène communiquant*	*SMA hétérogène non communiquant*

Dans chacun des cas cités dans le tableau on va définir la poursuite dans ce système, le cas général du système et les techniques pour réaliser la poursuite.

Tous les agents du système multi-agent de la solution sont des instances du modèle d'agent proposé dans la section précédente, un agent de quatre niveaux doté ou non de la propriété de communication.

Remarque *:* Une monde infini à deux dimensions est équivalent à un monde de cases, il suffit de faire tendre la taille de la case vers des valeurs très petites pour voir l'équivalence et l'inverse est correct il suffit de trouver l'epsilon physique dans le monde infini de 2D et créer un monde de cases avec taille epsilon.

3.1 Système Multi-agents homogène non communiquants

Tous les agents ont la même structure interne, buts, connaissance de domaine et des actions possibles. Ils ont également la même procédure pour la sélection de l'action. Les seules différences entre les agents sont leurs entrées sensorielles et les actions réelles prises de plus ils sont situés dans différentes positions dans le monde.

La structure d'un agent est de quatre niveaux, la structure interne est plus exactement le niveau physique, c'est-à-dire le contrôleur. Les contrôleurs de chaqun des types sont générés dans la morphogénéisation. On génère un premier contrôleur pour les poursuiveurs, et un deuxième pour les poursuivis, au cours de la poursuite les animats gardent le même contrôleur, d'où vient le caractère homogène de la poursuite.

a. Définition de la poursuite

Dans la poursuite avec un système multi-agent homogène non communiquant, on a un pour chaque poursuiveur un agent qui le contrôle, ces agent sont identique crées par le processus de morphogénéisation qui respecte l'architecture à quatre niveaux et chaque poursuivi et contrôlé par un agent du même type, avec la différence des buts et des objectifs qu'elle est exprimée au niveau Agent-Maker de l'architecture.L'homogénéité dans ce cas est liée au fait que les poursuiveurs ont le même contrôleur, et les poursuivis de mêmes.

Chaque agent poursuiveur choisit un poursuivi qui essaye de l'attraper avec les mêmes règles de capture que dans le cas de la poursuite de Cliff&Miller (voir Fig. 2). Le choix est fait au niveau comportemental par le critère du plus proche poursuivi. Cette extension peut être vue comme une multiplication par n du problème de Cliff&Miller, la seule différence c'est que les agents évoluent à chaque génération en faisant des sélections du meilleur.

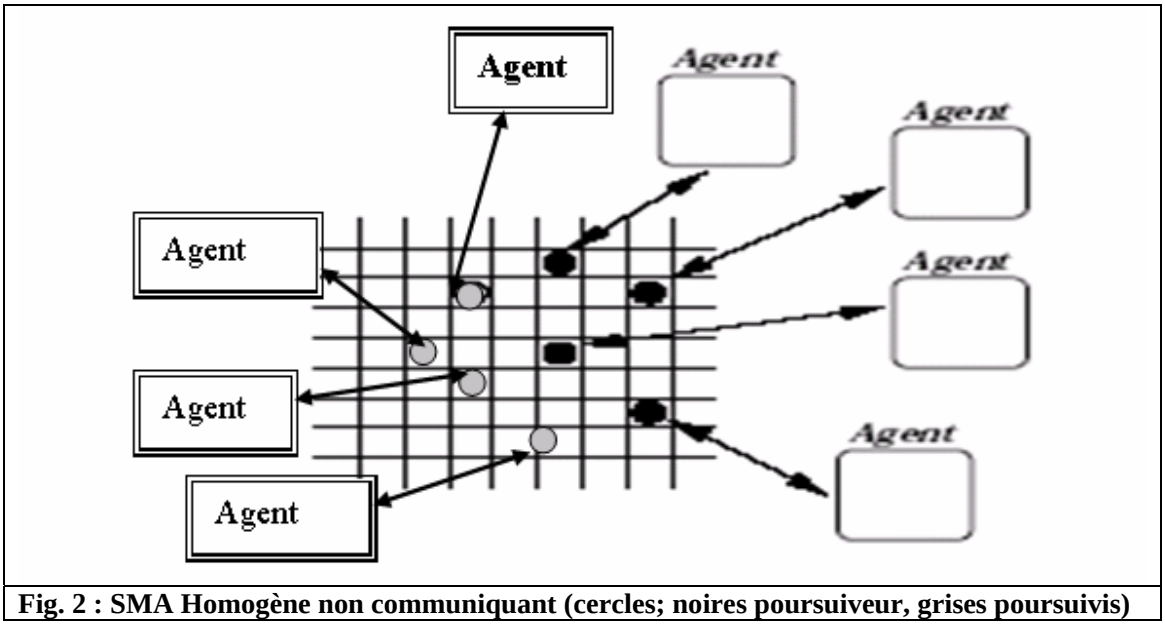


Fig. 2 : SMA Homogène non communiquant (cercles; noires poursuiveur, grises poursuivis)

b. Méthodes et résultats

Dans le modèle homogène non communiquant le déroulement de la poursuite se base sur quelques méthodes et fait apparaître quelques résultats :

Voici un tableau explicatif des méthodes qui peuvent être utilisées, ainsi que les techniques et les résultats retenues.

Les méthodes	Les résultats et les techniques
Agents réactifs vs. Agents délibératifs (réponses au changement monde)	- Comportement réactif pour l'entretien du groupe - Comportement délibératif pour la poursuite - Un mélange des comportements réactifs et délibératifs
Perspectives locales ou globales	- Une connaissance locale est meilleure parfois
Modélisation des états des autres agents de la poursuite	- Ne pas modéliser les autres, juste tirer l'attention
Comment affecter les autres agents de la poursuite (monde)	- La stigmergie (changement du monde pour affecter le groupe)

3.2 Système Multi-agents hétérogène non communiquants

Les agents n'ont pas la même structure interne, buts, connaissance de domaine et des actions possibles. Ils n'ont pas la même procédure pour la sélection de l'action et forcément pas les mêmes entrées sensorielles et les actions réelles prises de plus ils sont situés dans différentes positions dans le monde.

La structure d'un agent est de quatre niveaux, la structure interne est plus exactement le niveau physique, c'est-à-dire le contrôleur. Les contrôleurs de chaqun des types sont générés dans la morphogénéisation. On génère un premier contrôleur pour les poursuiveurs, et un deuxième pour les poursuivis, au cours de la poursuite les animats peuvent changer leurs contrôleur, d'où vient le caractère hétérogène de la poursuite.

a. Définition de la poursuite

Dans la poursuite avec un système multi-agent hétérogène non communiquant, on a un pour chaque poursuiveur un agent qui le contrôle. Ces agents ne sont pas identiques, crées par le processus de morphogénéisation qui respecte l'architecture à quatre niveaux et chaque poursuivi est contrôlé par un agent différent des autres. L'hétérogénéité dans ce cas est liée au fait que les poursuiveurs n'ont pas le même contrôleur, et de même pour les poursuivis.

Chaque agent poursuiveur choisit un poursuivi qui essaye de l'attraper avec les mêmes règles de capture que dans le cas de la poursuite de Cliff & Miller (voir Fig. 2). Le choix est fait au niveau comportemental par le critère du plus proche poursuivi. Le manque de communication dans ce cas peut causer des conflits entre les poursuiveurs, ceci rend la poursuite plus réelle. Pour résoudre les conflits on doit faire quelques changements sur les niveaux cognitif et comportemental (Voir section suivante).

Les mêmes problèmes der la poursuite qui émergent du système Homogène Non Communiquant sont dans le cas Hétérogène, en plus des problèmes liés à l'hétérogénéité qui sont la concurrence, compétition, conflits, gestion des ressources et d'autres.

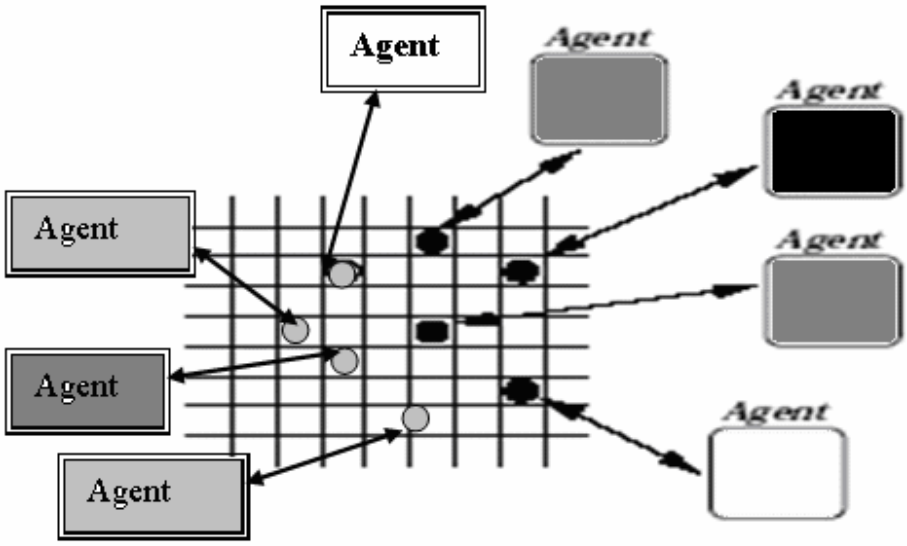


Fig. 3 : SMA Hétérogène non communiquant (cercles; noires poursuiveur, grises poursuivis)

b. Méthodes et résultats

Dans le modèle hétérogène non communiquant le déroulement de la poursuite se base sur quelques méthodes et fait apparaître quelques résultats :

Voici un tableau explicatif des méthodes qui peuvent être utilisées, ainsi que les techniques et les résultats retenus.

Les méthodes	Les résultats et les techniques
Coopération vs. Compétitivité entre les individus de la poursuite	- Théorie des jeux rectangulaires, jeu itératif. - Co-évolution Coopérative
Agents statique vs. apprenant (course aux armements)	- Co-évolution concurrentielle et compétitive
Modéliser les buts, actions, et connaissance des autres	- Déduire les intentions - capacités par l'observation. - Raisonnement Auto-épistémique - Raisonnement social : dépendre de

	d'autres pour le but
Gestion de ressources plus exactement l'énergie	- Apprentissage renforcé Multi-Agent pour l'équilibrage adaptatif de la charge.
Conventions sociales	- Conventions focales de points/émergent - Accords par des jetons de verrouillage
Rôles de chaque individu de la poursuite	- Agents remplissant différents rôles - Modèle en équipe (individu → rôle)

3.3 Système Multi-agents homogène communiquants

À ce point, nous n'avons pas encore considéré un SMA dans lequel les agents peuvent communiquer entre eux directement. Évidemment, la communication se fait via l'interaction de l'agent avec l'environnement. À l'aide de la communication, les agents (les poursuiveurs et les poursuivis) peuvent coordonner beaucoup plus efficacement. Cependant, la communication présente également plusieurs défis. Dans ce scénario, nous considérons les agents homogènes qui peuvent communiquer entre eux.

a. Définition de la poursuite

Dans le domaine de poursuite, la communication créée de nouvelles possibilités pour le comportement des poursuiveurs et des poursuivis. Cependant les agents peuvent échanger librement l'information afin de les aider à capturer le poursuivi plus efficacement pour les poursuiveurs et de s'échapper des poursuiveurs pour les poursuivis. La situation actuelle est illustrée sur la Fig. 4.

Chaque agent poursuiveur choisit un poursuivi qui essaye de l'attraper avec les mêmes règles de capture que dans le cas de la poursuite de Cliff & Miller. Le choix est fait au niveau comportemental par le critère du plus proche poursuivi. Grâce à la communication les agents poursuiveurs communiquent leurs positions et même la position de leurs poursuivis et d'autre informations comme l'énergie consommé, la vitesse du maximale et la position du plus proche poursuivi.

Ceci rend la poursuite plus réelle, mais rend les conflits plus délicats, comme les inter-blocages, les collisions et la limitation du champ de vision, pour résoudre ces problèmes on doit faire quelques changements sur la boucle de génération des agents « le processus de morphogénéisation ». Il faut pour cela pénaliser les agents qui provoquent des inter-blocages et des collisions. Ces changements sont effectués dans la fonction objective de génotype qui a généré l'agent (poursuiveur ou poursuivi). Cette modification de la fonction objective va éliminer les mauvais individus de la sélection pour les prochaines générations.

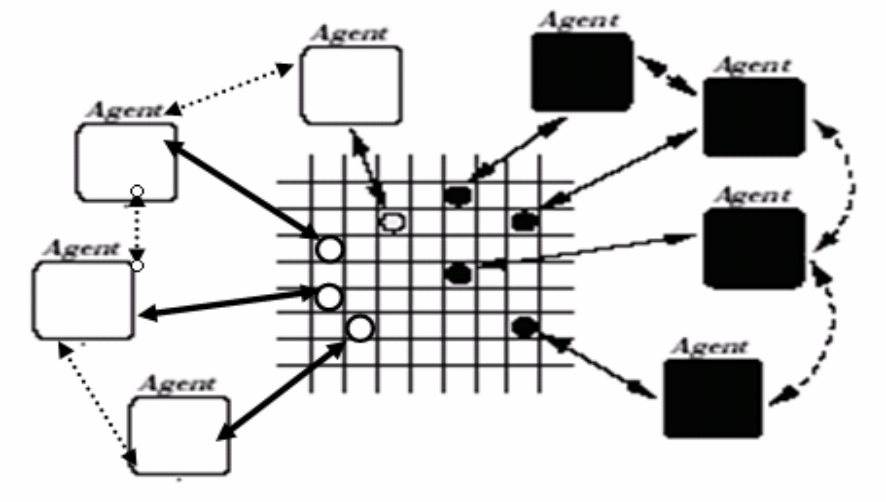


Fig. 4 : SMA Homogène communiquant (cercles; noires poursuiveur, blanches poursuivis)

b. Méthodes et résultats

Dans le modèle homogène communiquant le déroulement de la poursuite se base sur quelques méthodes et fait apparaître quelques résultats :

Voici un tableau explicatif des méthodes qui peuvent être utilisées, ainsi que les techniques et les résultats retenus.

Les méthodes	Les résultats et les techniques
Sensation distribuée des individus dans le monde de la poursuite	- Sensation active du monde - Propagation de requêtes pour trouver les chemins emprunté par un individu (poursuivi par Exp.) - Le produit des visions des individus donne un mapping complet du monde de la poursuite
Le contenu de la communication entre les individus	- La communication de l'état (positions, vitesses …) - La communication des buts (lieu de capture …)

3.4 Système Multi-agents hétérogène communiquants

Le scénario examiné dans la section 3.2, les agents diffèrent dans plusieurs propriétés comme leur entrées sensorielles, leurs buts, leurs actions, et leur connaissance de domaine. De tels systèmes multi-agent hétérogènes peuvent être très complexes et puissants. Cependant le plein pouvoir de SMA peut être réalisé en ajoutant capacité pour

que les agents hétérogènes communiquent entre eux. En fait, combinant la communication et l'hétérogénéité présente la possibilité d'avoir un système multi-agent qui fonctionne pareillement à un simple-agent système en termes de commande. En envoyant leurs entrées de capteurs à et en recevant leurs commandes d'un agent, tous les autres agents peuvent rendre la commande à cet agent simple. Dans ce cas-ci, la commande n'est plus distribuée.

a. Définition de la poursuite

Chaque agent poursuiveur choisi un poursuivi qui essaye de l'attraper avec les mêmes règles de capture que dans le cas de la poursuite de Cliff&Miller. Le choix est fait au niveau comportemental par le critère du plus proche poursuivi.

Tenir compte de l'hétérogénéité et de la communication dans le domaine de poursuite ouvre de nouvelles possibilités de la commande. La situation actuelle est illustrée sur la figure 5. On peut employer des agents communiquant dans le domaine de poursuite pour entreprendre quelques expériences intéressantes d'apprentissage multi-agent. Dans l'instanciation du domaine, il y a plusieurs agents poursuivis et plusieurs agents poursuiveurs avec une vision limitée de sorte qu'ils ne puissent pas toujours savoir où les poursuivis sont. Ainsi les poursuiveurs peuvent s'entraider en s'informant de leur entrée sensorielle. Il a été montré que les poursuiveurs pourraient également s'aider en échangeant des épisodes de renfort et/ou commander des politiques même cas pour les poursuivis qui peuvent communiquer des stratégies, des politiques et même des scénarios d'échappement.

Dans une présentation complète du domaine de poursuite, on considère également la gamme complète des possibilités de communication. On considère que tous les poursuiveurs (resp. poursuivis) peuvent échanger des données, des buts, deux à deux. La proportion entre les coûts inférieurs de communication et les meilleures décisions sont décrites comme une largeur de bande et une consommation limitée de temps de calcul.

Une manière de borner cette proportion est de faire en sorte que l'augmentation de coût de communication (temps) diminue la liberté de la décision de l'agent poursuiveur(resp. poursuivi). On suggère que les poursuiveurs (resp. poursuivis) devraient se déplacer dans quatre phases, dans l'ordre croissant du coût (liberté décroissante), les quatre phases sont: autonomie, communication, négociation, et commande . Quand les poursuiveurs (resp. poursuivis) cessent d'accomplir le progrès suffisant vers les poursuivis(resp. loin des poursuiveurs) en utilisant une stratégie, ils devraient se déplacer à la prochaine stratégie la plus chère. Ainsi les poursuiveurs (resp. les poursuivis) peuvent se rapprocher (resp. s'éloigner) efficacement et effectivement des poursuivis (resp. poursuiveurs).

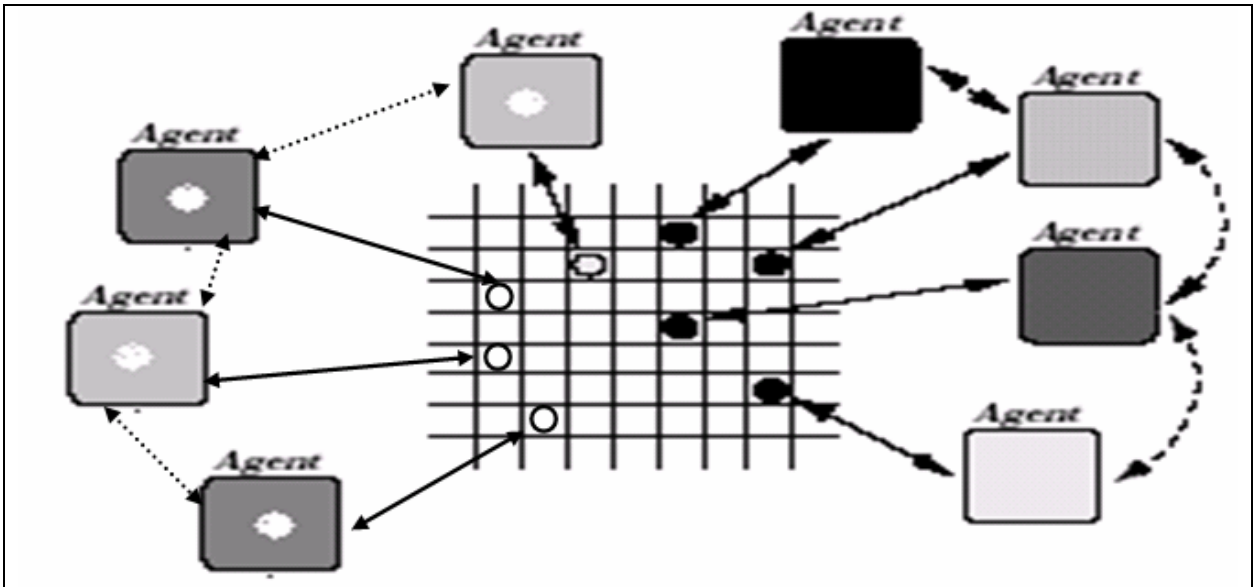


Fig. 5 : SMA Hétéro communiquant (cercles; noires poursuiveur, blanches poursuivis)

b. Méthode et résultats

Dans le modèle hétérogène communiquant le déroulement de la poursuite se base sur quelques méthodes et fait apparaître quelques résultats :

Voici un tableau explicatif des méthodes qui peuvent être utilisées, ainsi que les techniques et les résultats retenues.

Les méthodes	Les résultats et les techniques
Se comprendre	- Protocoles et langages : KIF, KQML, frais. - Fondant la signification par l'intermédiaire d'une expérience partagée. - Intégration de systèmes de legs. - Connaissance des langues.
Actes communicatifs de planification	- Actes de la parole. - Comportements sociaux d'étude. - Raisonnement au sujet de l'exactitude.
Actes communicatifs d'Apprentissage	- Q-apprentissage des Multi-Agent. - Formation des Q-fonctions d'autres agents (voie conduisant). - Réduire au minimum le besoin de formation. - Co-évolution coopérative.
Bienveillance contre	- Filets de contrat pour le commerce électronique.

la compétitivité	- Systèmes basés sur le marché. - Etude bayésienne dans la négociation : modèle d'autres. - Méthodes basées sur le marché pour des contraintes distribuées.
Négociation	
Gestion de ressource	- Planification globale partielle généralisée (GPGP). - Apprenant à choisir parmi des méthodes de coordination. - Réponse de question dans des réseaux de l'information. - Division des tâches indépendantes.
Engagement / Non-engagement	- Engagements internes, sociaux, et collectifs (de rôle). - Etats d'engagement (potentiel, pré, et réel) comme états de planification. - Modèle de Belief/desire/intention (BDI) : OASIS.
Localisation de collaboration	- Coalitions. - Données incertaines fondantes de sonde.

4. Conclusion

Dans ce chapitre on a essayé de donner une architecture nouvelle et hybride pour des agents de la poursuite « COCAGE ». Ces agents sont dotés d'une puissance d'expression et de génération de leurs actions, leurs intensions et même de leurs morphologies, tout ça dans une première partie. Dans la deuxième partie on est intéressé à l'infrastructure du système de la poursuite une notion qui n'existe pas dans les anciens modèles de la poursuite, dans ce système il y a deux facteurs principaux qui sont l'Homogénéité des agents, et leur Communication. Ce qui a donné quatre systèmes de poursuite, dans chaque système on a donné les techniques et méthodes qui peuvent être utilisées pour réaliser la poursuite définie.

Cette infrastructure s'inspire de tous les travaux qui existent dans les domaines de la planification, la sensation, la négociation, comportement, raisonnement automatique et d'autres. Le système de poursuite est ouvert pour n'importe quelle extension du point de vue agent et même du point de vue architecture. Les agents sont en évolution perpétuelle et l'architecture du système peut changer grâce à la possibilité d'émergence (génétique) de la communication.

Conclusion Générale

Le problème de la poursuite de Cliff & Miller continue à avoir plus d'importance dans le domaine de la simulation des systèmes artificiels, et à susciter l'intérêt de beaucoup de chercheurs, que ce soit pour plus de tests ou pour faire des extensions comme c'était notre cas. Le point essentiel dans ces simulations est dans la co-évolution des espèces en compétition, plusieurs chercheurs voient la co-évolution comme un paradigme de programmation et commencent à utiliser cette idée pour trouver des algorithmes pour des problèmes usuels comme la recherche, le tri et autres.

Notre extension *COCAGE,* s'inscrit dans la catégorie des systèmes distribués pour la résolution des problèmes, en utilisant. On même temps la co-évolution collective dotée généralement d'une coopération entre agents du système. Cette extension est plus une approche qu'une infrastructure, grâce à la structure générique de l'agent et l'ouverture complète du système Multi-Agents. COCAGE peut être utiliser dans le problème de la poursuite de Cliff & Miller comme dans n'importe quel problème de nature distribuée où plusieurs espèces sont en co-évolution pour réaliser un but quelconque.

L'approche COCAGE englobe l'algorithme génétique qui crée les individus, l'interpréteur neuronal, le monde de la poursuite, le sélecteur d'actions, planificateur de tâches, les communications entre agents, évaluateur de générations et enfin les propriétés du système multi-agents. On voit que l'approche est autonome du point de vue de structure et comportement. Les agents comme présentés dans le chapitre VII, sont assez autonomes pour qu'ils puissent évoluer, changer de structure, activer la communication, modifier le comportement et planifier les tâches provenant du niveau réactif. Une

autonomie de ce niveau est toujours souhaitable pour la résolution des problèmes, surtout dans le cas où les problèmes sont aussi de nature distribuée.

Une des perspectives les plus remarquables est d'utiliser l'approche pour des problèmes voisins comme le ramassage de miettes, Multi-robot Football Game, les contrôleurs des missiles anti-missiles avec plusieurs têtes, le routage des paquets dans les systèmes distribués. Une autre perspective, est de choisir une des infrastructures des agents pour doter cette dernière par l'architecture d'agent proposée dans COCAGE, mais ça demande plus de travail à cause de la fermeture des infrastructures en ce qui concerne la morphologie et le comportement des agents. Quand même on trouve la plateforme MADKIT qui permet d'écrire quelques scripts Java pour le comportement mais rien pour la morphologie. L'implémentation d'une infrastructure agent pour l'approche COCAGE peut nécessiter une re-programmation complète d'une infrastructure parmi celles qui existent, qui commence de la codification de contrôleurs neuronaux jusqu'à la communication des politiques de résolution de problèmes.

La bibliographie

[AGHA,1986] AGHA.G “Actors : A model of Concurrent Computation for Distributed Systems”. MIT Press, 1986.

[Ahuactzin et al. , 1995] Ahuactzin J-M., Mazer E. et Bessière P., *Fondements mathématiques de l'algorithme "fil d'Ariane"*, Revue d'Intelligence Artificielle, 1995.

[Alliot et Schiex, *1993*] Alliot J.M. and T. Schiex (*1993*) : ``*Intelligence artificielle et informatique théorique*", Cépadues Editions.

[Anderson,1990] T.L. Anderson and M. Donath, Animal Behavior as a Paradigm for Developing Robot Autonomy, in [Maes90] , pp. 145-168, 1990.

[Andreoni et Miller, *1995*] Andreoni J. and J. Miller (*1995*) : ``*Auctions with Artificial Adaptative Agents*", Games and Economic behavior, Vol. 10, pp 39-64.

[Angeline et al. ,1994] Peter J. Angeline, Gregory M. Saunders, and Jordan B. Pollack. An algorithm that constructs recurrent networks, 1994. evo-lutionary

[Arifovic , *1994*] Arifovic J. (*1994*) : `` *Genetic algorithm learning and the cobwell model*", Journal of Economic Dynamic and Control, No 18, pp.2-28.

[Arifovic et Eaton ,*1995*] Arifovic J. and C. Eaton (*1995*) : `` *Coordination via Genetic Learning*", Computational Economics, Vol.8, No. 3, pp. 181-203.

[Arthur,*1991*] Arthur W. B. (*1991*) : ``*Designing Economic Agents that Act Like Human Agents : A Behavioral Approach to Bounded Rationality*", AEA Papers and proceedings, Vol. 81, No.2.

[Astor,1998] J. C. Astor. A developmental model for the evolution of artificial neural networks: Design, implementation, evaluation. Master’s thesis, of Heidelberg, September 1998. Univer-sity

[Attoui,1997] Attoui.A “Un système multi-agents a temps réel”.Edition Eyrolles 1197.

[Axelrod , *1987* Axelrod R. (*1987*) : ``*The Evolution of Strategies in the Iterated Prisoner's Dilemma''*, in Genetic Algoritms and Simulated Annealing, L. Davis editor, Pitman, London.

[Baluja , 1998] Baluja S., *Finding Regions of Uncertainty in Learned Models : An Application to Face Detection.* 5th Conference on Parallel Problem Solving from Nature, Springer Verlag, pp. 663-671, Amsterdam, Pays-Bas, 1998.

[Baujard,1992] Baujard.O “Conception d’un environnement de développement pour la résolution de problèmes:Apport de l’intelligence artificielle et application à la vision”. Thèse de l’université de Joseph Fourier. Grenoble I, 1992

[Beaumont et Bradshaw ,1995] Beaumont P. and P. Bradshaw (*1995*) : `` *A distributed Parallel Genetic algorithm for solving optimal Growth Models*", Computational Economics, Vol.8, No. 3, pp. 159-179.

[Beer ,1995] R.D.Beer. On the dynamics of small continuous-time recurrent neural networks, *Adaptive Behavior*, 3(4):471-511,1995.

[Beer, 1990] R.D. Beer, H.J. Chiel and L.S. Sterling, A Biological Perspective on Autonomous Agent Design, in [Maes90] , pp. 169-186, 1990.

[Belew , 1989] Belew R., *When Both individuals and populations search : Adding simple learning to the genetic algorithm*. Proceedings of the Third Conference on Genetic Algorithms. Morgan Kaufmann [San Mateo,1989.]

[Belew et al. , 1990] Belew B., McInerney J. et Schraudolph N., *Evolving Networks : Using the Genetic Algorithms with Connectionist Learning.* CSE Technical Report CS90-174, Computer Science, UCSD, 1990.

[Birchenhall , 1995] Birchenhall C. (*1995*) : ``*Introduction of Computational Economics special issue on Genetic algorithm''*, Computational Economics, Vol.8, No. 3, pp. 155-158.

[Birchenhall ,1995a] Birchenhall C. (*1995*) : ``*Modular technical change and Genetic algorithms''*, Computational Economics, Vol.8, No. 3, pp. 233-253.

[Boers et al. ,1993] E. J. W. Boers, H. Kuiper, B. L. M. Happel, and I. G. Designing modular artificial neural networks. Technical Springhuizen-Kuiper. report, 1993.

[Braitenberg 1984] V. Braitenberg, *Vehicles, Experiments in Synthetic Psychology*, MIT Press Bradford Books, Cambridge MA,1984

[Brooks, 1991] R.A. Brooks, Intelligence Without Reason, IJCAI-91, Proceedings of the twelfth International Joint Conference on Artificial Intelligence, Vol 1 (1991), pp. 569-595.

[Brooks,1986] Brooks R.A. “A robust layered control system for a mobile robot”. IEEE Journal of Robotics and Automation, 2(1) :14-23, 1986.

[Brouard et al. , 1998] Brouard T., Slimane M., Venturini G. et Asselin De Bauville J-P., *Apprentissage génétique hybride de chaînes de Markov cachées*. Apprentissage : des principes naturels aux méthodes artificielles, ed. Ritschard G., Berchtold A., Duc F.et Zighed D., éditions Hermès, pp.241-256, 1998.

[Carlos et al,1997] Joao Carlos, Figueira Pujol and Riccardo Poli. Evolving neural controllers using a dual representation. Technical Report CSRP- 97- 25, School of Computer Science, University of Birmingham, september 1997.

[Catoni ,1990] Catoni O. (*1990*) : ``*Large deviations for Annealing*", Thèse de Doctorat, Université de Paris XI..

[Cerf ,1994] Cerf R. (*1994*) : ``*Une théorie asymptotique des algorithmes génétiques*", These de Doctorat, Université de Montpellier II.

[Chaib,1994] CHAIB-DRAA.B " Coordination between agents in routine, familiar and unfamiliar situations". Rapport de recherche DIUL-RR-9401 du Département d'informatique, Université Laval, Ste-Foy, PQ, Canada, 1994.

[Cliff ,1994] D. Cliff. NCAGE: Network control architecture genetic encoding; Technical Report CSRP 325, University of Sussex School of Cognitive and Computing Sciences, 1994.

[Cliff et al, 1993] D.Cliff, I. Harvey and P. Husbands. Explorations in evolutionary robotics, *Adaptive Behavior*,2(1):71-108,1993.

[Cliff et al, 1996] D.Cliff, I. Harvey and P. Husbands. Artificial evolution of visual control systems for robots. In M. Srinivisan and S. Venkatesh. editors, *From Living Eyes to Seeing Machines*. Oxford University Press1996 .

[Cliff et Miller, 1995] D. Cliff and G. F. Miller. Tracking the Red Queen_ Measurements of adaptive progress in co-evolutionary simulations. In F. Moràn. A. Moreno. J. J. Merelo. and P. Chacon, editors, *Advances in Artificial Life, Proc. Third Euro. Conf. Artificial Life*, pp. 200-218 Springer-Verlag, 1995

[Cybenko, 1988] G. Cybenko. Continuous valued neural networks with two hidden layers are sufficient. Technical report, Departement of computer science, Tufts University, Medford, MA, 1988.

[Cybenko, 1989] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signal and System*s, (2): 303– 314, 1989.

[De Jong , 1975] De Jong, K., *An analysis of the behavior of a class of genetic adaptive systems*. Ph.D. thesis, University of Michigan, 1975.

[Della,2000] Della.R "Une plate-forme de développement de systèmes multi-agents mobiles coopératifs sur Internet/Intranet" Thèse de Magistère, USTHB 2000.

[Dorsey et Mayer ,*1995*] Dorsey R. E. and W. J. Mayer (*1995*) : ``*Genetic Algorithms for Estimation Problems With Multiple Optima, Nondifferentiability, and Other Irregular Features*", Journal of Business and Economic Statistics, Vol.13, No 1.

[Durand et al , 1994] Durand N., N.Alech, J. M. Alliot and M. Shoenauer (*1994*) : ``*Genetic Algorithms for Optimal Air Trafic Conflict Resolution*" In Proceedings of the second Singapore conference on Intelligent Systems, SPICIS.

[Eiben et al. , 1998] Eiben A.E., van der Hauw J.K. et van Hemert J.I., *Graph Coloring with Adaptive Evolutionary Algorithms*. Journal of Heuristics, vol. 4:1, pp. 25-46, 1998.

[Erman al,1980] Erman L. D., F.Hayes R., Lesser V.R., and Reddy R. D. " The hearsay-II speech understanding system : Integrating knowledge to resolve uncertainty". ACM Computing Surveys, 12(2), 1980.

[Farley et Jones ,*1994*] Farley A. M. and S. Jones (*1994*) : ``*Using a Genetic Algorithm to determine an Index of Leading Economic Indicators*", Computational Economics, Vol. 7, No 3.

[Ferber et Carl,1991] Ferber.J ,Carl.P "Actors and agents as reflective concurrent object : a MeringIV perspective". IEEE transaction on systems, man and cybernetics,21(6),1991

[Ferber et Ghallab ,1988] FERBER J.,GHALLAB M. « Problématique des univers multi-agents ». Actes des journées nationales du PRC-IA. Mars 1988

[Ferber,1989] Ferber.J "Objet et agent: une étude de structures de représentation et de communication en intelligence artificielle" Thèse de l'université ParisIV, Juin 1989.

[Ferber,1995] Ferber.J "Les SMA: vers une intelligence collective",InterEditions, Paris 1995.

[Fontanili et Vincent , 1997] Fontanili F. et Vincent A., *Comment optimiser le fonctionnement d'un atelier avec la simulation de flux ?* Revue Française de Gestion Industrielle, vol. 16, n. 3, 1997.

[Freidlin et Wentzell ,*1983*] Freidlin M. I. and Wentzell (*1983*) : ``*Random Perturbations of Dynamical Systems*", Springer Verlag, New-York.

[Fullmer et Miikkulainen, 1991] Brad Fullmer and Risto Miikkulainen. Using marker- based genetic of neural networks to evolve finite- state behaviour. In *Proceedings* en-coding *of the first European Conference on Artificial Life (ECAL- 9*1, Paris, 1991.

[Galliers,1988] GALLIERS J.R. "A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflit". PhD thesis, Open University, UK, 1988.

[Gasser,1990] Gasser.L "Social Conceptions of knowledge and action". Technical report ACT-AI-355-90,MCC, October 1990.

[Geyer et al. , 1998] Geyer H., Ulbig P. et Schulz S., *Encapsulated Evolution Strategies for the Determination of Group Contribution Model Parameters in Order to Predict Thermodynamic Properties*. 5th Conference on Parallel Problem Solving from Nature, Springer Verlag, pp. 663-671, Amsterdam, Pays-Bas, 1998.

[Goffe et al ,1994] Goffe L., Ferrier G. D. and J. Roger (*1994*) : ``*Global optimization of statistical functions with simulated annealing*", Journal of Econometrics, Vol. 60, pp. 65-99.

[Goldberg , 1989] Goldberg D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Massachusetts, 1989.

[Goldberg et Richardson, 1987 D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *of the Second International Conference on Genetic Algorithm*s, *Pro-ceedings* pages 148– 154, San Francisco, 1987. CA: Morgan Kaufmann.

[Goldberg, 1989] Goldberg D. (*1989*) : ``*Genetic Algorithms*" Addison Wesley editor.

[Grefenstette , 1992] J. J. Grefenstette. The evolution of strategies for multi-agent behaviours, *Adaptive Behavior*, 1(1):65-89,1992$

[Gruau et al, 1996] Frederic Gruau, Darrell Whitley, and Larry Pyeatt. A comparison cellular encoding and direct encoding for genetic neural networks. be-tween In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. editors, *Genetic Programming 1996: Proceedings of the First Annual* Ri-olo, *Conferenc*e, pages 81– 89, Stanford University, CA, USA, 28– 31 1996. MIT Press.

[Gruau et Whitley , 1993] Gruau F. et Whitley D., *Adding Learning to the Cellular Development of Neural Networks : Evolution and the Baldwin Effect.* Evolutionary Computation 1[3: 213-233, 1993.]

[Gruau, 1992] Fédéric Gruau. Cellular encoding of genetic neural networks. Technical Report 9221, Laboratoire de l'informatique du Parallélisme, Ecole Supérieure de Lyon, 1992.

[Gruau, 1994] F. Gruau. *Synthèse de Réseaux de Neurones par Codage Cellulaire et Algorithmes Génétique*s. PhD thesis, ENS Lyon, Université Lyon I, 1994.

[Harvey, 1993] Inman Harvey. *The Artificial Evolution of Adaptive Behavio*r. PhD thesis, University of Sussex, 1993.

[Haton et al,1991] HATON J. P., BOUZID N., CHARPILLET F., HATON M. C., LAASRI B., LAASRI H., MARQUIS P., MONDOT T., NAPOLI A. "Le raisonnement en intelligence artificielle (modèles, techniques et architectures pour les systèmes à base de connaissances". InterEdition 1991. Paris.

[Heap et Samaria , 1995] Heap T. et Samaria F., *Real-Time Hand Tracking and Gesture Recognition Using Smart Snakes*. Technical Report 95.1 AT&T Laboratories Cambridge, 24a Trumpington Street, Cambridge CB2 1QA, England, 1995.

[Herrnstein, 1991] Herrnstein R. J. (*1991*) : ``*Experiments on stable Suboptimality in Individual Behavior'', AEA Papers and proceedings*", Vol. 81, No.2.

[Hewitt et al,1973] Hewitt .C,Bishop.P,Steiger.R. "A universal modular actor formalism for artificial intelligence" In third international joint conferance on artificial intelligence, 1973.

[Hewitt,1991] Hewitt C. "Open Information Systems Semantics for Distributed Artificial Intelligence", Artificial Intelligence 47(1-3)pp 79-106 1991.

[Holand et Miller, 1991] Holland J. H.and J. H Miller (*1991*) : ``Artificial *Adaptation Agents in Economic Theory*", American Economic Review papers Proceedings, Vol. 81, no 2.

[Holland ,1975] Holland J. H. (*1975*) : ``*Adaptation in Natural and Artificial Systems*" , University of Michigan press.

[Iwata et al. , 1990] Iwata M., Kitani I., Yamada H., I ba H. et Higuchi T., *A Pattern Recognition System Using Evolvable Hardware*. 1st Conference on Parallel Problem Solving from Nature, Berlin, Allemagne, 1990.

[Jenning et al,1998] Jennings.N, Wooldridge.M, Sycara.K "Application of intelligent agent". in Agent Technology : Foundations, Applications, and Markets, N. Jennings and M.

Wooldridge, eds. Springer Verlag. 1998.

[Kane et Schoenauer , 1997] Kane C. et Schoenauer M., *Optimisation Topologique de Formes par Algorithmes Génétiques*. Revue Française de Mécanique 4, pp 237- 246, 1997.

[Kenneth et al , 200]1 Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. Technical Report TR- AI- 01- 290, of Texas at Austin University, June 2001.

[Khoaldi,1994] Khoualdi . "Filtrage d'alarmes pour un système automatisé par une approche multi-agents " Thèse de l'université Paris VI. LAFORIA TH94/07. 18 Novembre 1994.

[Kitano,1990] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex System*s, 4: 461– 476, 1990.

[Kodjabachian 1998 J. Kodjabachian. Développement et évolution de réseaux de neurones artificiels. PhD thesis, Université Pierre et Marie Curie, 1998.

[Kodjabachian et Meyer, 1995] J. Kodjabachian and J. A. Meyer. Evolution and development of control architectures in animats. *Robotics and Autonomous System*s, 16: 161– 182, 1995.

[Kodjabachian et Meyer, 1997] J. Kodjabachian and J. A. Meyer. Evolution and development of networks controlling locomotion, gradient- following, and neu-ral in artificial insects. *IEEE Transactions on Neural Network*s, obstacle-avoidance 9: 796– 812, 1997.

[Kodjabachian, 1998] J. Kodjabachian. *Développement et évolution de réseaux de neurones artificiel*s. PhD thesis, Université Pierre et Marie Curie, 1998.

[Koza, 1992] John R. Koza. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.

[Labdi et Lejouad,1993] LABIDI.S, LEJOUAD.W "Form Distributed Artificial Intelligence to Multi-agent Systems". INRIA Sophia Antipolis, August 1993.

[Land et Colett,1974] M. F. Land and T. S. Collett. Chasing behaviour of house flies,(*Fannia canicularis*). *Journal of Comparative Physiology*, 89:331-357, 1974

[Lapedes et Farber, 1987] A. Lapedes and R. Farber. *Neural Information Processing System*s, How neural nets work. New York: American Institute of Physics, chapter 1987.

[Lindenmayer, 1968] Aristid Lindenmayer. Mathematical models for cellular interaction in parts i and ii. *Journal of theoretical biology*, (18): 280– 315, development, 1968.

[Liu et Wechsler , 1998] Liu C. et Wechsler H., *Face Recognition Using Evolutionary Pursuit*. Fifth European Conference on Computer Vision, ECCV'98, Université de Freiburg, Allemagne, juin 1998.

[Luke et Spector, 1996] Sean Luke and Lee Spector. Evolving graphs and networks with edge encoding: Preliminary report. In *Late Breaking Papers of the Genetic Programming 96 (GP96*), 1996.

[Maes, 1989] P. Maes, The dynamics of action selection, in: N. Sridharan, editor, Proc. 11th IJCAI, pp. 991-997, 1989.

[Meyer et al, 1991] Meyer, J.A. & Wilson, S.W. (Eds). *From Animals to Animats. Proceedings of the First International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: The MIT Press/Bradford Books. 1991.

[Michalewicz ,*1991*] Michalewicz Z. (*1991*) : `` *Genetic Algorithms + Data Structures = Evolution programs"* Springer Verlag, New-York.

[Michel et al. ,1997] Olivier Michel, Manuel Clergue, and Philippe Collard. Artificial Applications to the cart- pole problem and to an autonomous neu-rogenesis: mobile robot, 1997.

[Miller et Cliff, 1994] G. F. Miller and D. Cliff. Protean behavior in dynamic games. Arguments for the co-evolution of pursuit-evasion tactics. In D. Cliff, P. Husbands, J.A. Meyer, and S. Wilson editors. *From Animals to Animats 3. Proc. Third Int. Conf. Sim. Adapt. Behav., (SAB94)*,pp. 411-420. MIT Press.1994

[Miller et Freyd, 1993] G. F. Miller and J. J. Freyd. Dynamic mental representations of animate motion. The interplay among evolutionary cognitive, and behavioral dynamics. Technical Report CSRP 290. University of Sussex School of Cognitive and Computing Sciences, 1993.

[Miller, 1996] G. F. Miller. Protean primates, The evolution of adaptive unpredictability in competition and courtship. In A. Whiten and R. W. Byrne. Editors, *Machiavellian Intelligence II* Oxford University Press 1996

[Oussedik et Delahaye , 1998] Oussedik S. et Delahaye D., *Reduction of Air Traffic Congestion by Genetic Algorithms*. 5th Conference on Parallel Problem Solving from Nature, Springer Verlag, Amsterdam, Pays-Bas, 1998.

[Paechter et al.,1998] Paechter B., Rankin R.C., Cumming A. et Fogarty T.C., *Timetabling the Classes of an Entire University with an Evolutionary Algorithm*. 5th Conference on Parallel

Problem Solving from Nature, Springer Verlag, Amsterdam, Pays-Bas, 1998.

[Pasemann et Dieckmann, 1997] Frank Pasemann and Ulrich Dieckmann. Evolved neurocontrollers for pole-balancing. In *Biological and Artificial Computation: From to Technology. IWANN'97*, 1997.

[Payton, 1990] D.W. Payton, Internalized Plans: A Representation for Action Ressources, in [Maes90] , pp. 89-103, 1990.

[Petit-Singeot et Cazoulat ,*1994*] Petit-Singeot F. and R. Cazoulat (*1994*) `` Réplicateurs Adaptatifs et Théorie des Jeux" , in Journées Evolution Artificielle, 20-23 septembre 94, ENAC.

[Petter et De Jong, 1]994 M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor and H. P. Schwefel, editors. *Proc. Third Conference on Parallel Problem Solving from Nature.*pp. 249-257. Springer-Verlag. 1994

[Pujol et Poli, 1997] Joao Carlos Figueira Pujol and Riccardo Poli. Evolving neural controllers using a dual representation. Technical Report CSRP- 97- 25, School of Computer Science, University of Birmingham, september 1997.

[Reynolds, 1994] C. Reynolds. Competition- coévolution, and the game of tag. In R. Brooks and P. Maes. Editors. *Artificial Life IV*, pp.59-69. MIT Press1994

[Roux et Jacq , 1993] Roux C. et Jacq J.J., *Registration of successive DSA images using a simple genetic algorithm with a stochastic performance function*. Proceeding of 19th IEEE Northeast Bioengineering Conf., Newark NJ, 223-224, 1993.

[Rumelhart et McClelland, 1986] D. E. Rumelhart and J. L. McClelland. Parallel Distributed Processing. Explorations in the Microstructure of Cognition. The MIT Press/ Bradford Books, 1986.

[Schultz et Grefenstette , 1992] Schultz A.C. et Grefenstette J.J., *Using a Genetic Algorithm to Learn Behaviors for Autonomous Vehicles*. AIAA Guidance, Navigation and Control Conference, Hilton Head, SC, 1992.

[Shoham,1993] Shoham.Y "AGENT0: A simple agent language and its interpreter". In Proceeding of the ninth National Conference on Artificial Intelligence(AAAI91), USA 1993.

[Sims, 1994] K. Sims. Evolving .3D morphology and behavior by competition. In R. Brooks and P. Maes. Editors. *Artificial Life IV*. pp. 28-39.MIT Press 1994.

[Stanley et Miikkulainen ,2001] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. Technical Report TR- AI- 01- 290, of Texas at Austin University, June 2001.

[Stork et al, 1992] D. Stork. B. Jackson. and S. Walker. Non-Optimality via pre-adaptation in simple neural systems. In C. Langton. C. Taylor, J. D. Farmer, and S. Rasmussen, editors. *Artificial Life II*.pp 409-429. Addison Wesley 1992.

[Sycara,1989] Sycara K "Multiagent Compromise via Negotiation". In Distributed Artificial Intelligence, Vol. 2, L. Gasser and M. N. Huhns (eds.), Morgan Kaufmann Publishers, Los Altos, CA, 1989, pp. 119-137.

[Takagi et Sugeno, 1985] T. Takagi and M. Sugeno. Fuzzy identification of systems and its to modelling and control. In *IEEE Transactions on Systems*, appli-cation *Man and Cybernetic*s, volume 15. 1985.

[Tao et Michalewicz , 1998] Tao, G. and Michalewicz Z., *Inver-over Operator for the TSP*, Proceedings of the 5th Parallel Problem Solving from Nature, Springer-Verlag, Lecture Notes in Computer Science, Amsterdam, Septembre 1998.

[Tordjman ,*1994*] Tordjman H. (*1994*) : `` *Dynamiques spéculatives, hétérogénéité des agents et apprentissage : Le cas des taux de change*", These de Doctorat, C.E.F.I;, Université d' Aix Marseille II.

[Trouvé ,*1993*] Trouvé A. (*1993*) : ``*Parallélisation massive du recuit simulé"*, Thèse de Doctorat, Université de Paris XI.

[Tyrrel, 1993] T. Tyrrell, Computational Mechanisms for Action Selection, PhD Thesis, University of Edinburgh, 1993.

[Vincent,1993] VINCENT C. "Etude et mise en œuvre du paradigme multi-agents de ATOME A GTMAS", Thèse de Doctorat de l'université de Nancy 1, France 1993.

[Vriend ,*1995*] Vriend N. J. (*1995*) : `` *Self-Organisation of Markets : An Example of a Computational Approach*", Computational Economics, Vol.8, No. 3, pp. 205-231.

[Watson et al. , 1998] Watson J.P., Ross C., Eisele V., Denton J, Bins J., Guerra C., Whitley D. et Howe A., *The Traveling Salesrep Problem, Edge Assembly Crossover, and 2-opt*. 5th Conference on Parallel Problem Solving from Nature, Springer Verlag, Amsterdam, Pays-Bas, 1998.

[Werner et Dyer,1993] G. M. Werner and M. G. Dyer. Evolution of herding behaviours in artificial

animals. In J.A. Meyer, H. Roitblat, and S. Wilson, editors, *Proc. Second Int. Conf. Sim. Adapt. Behav.(SAB92)*.pp393-399 MIT Press 1993

[Whitley et al ,1995] D. Whitley, F. Gruau, and L Pyeatt. Cellular encoding applied to In *Siwth International Conference on Genetic Algorithm*s, neuro-control. 1995.

[Wooldrige et Jennings,1994] Wooldridge.M, Jennings.N "Towards a theory of cooperative problem solving" Damazeau.Y, Muller.J-P, Parram.J (Ed).Odense,Danemark,1994.

[Wooldrige et Jennings,1995] Wooldridge.M, Jennings.N "Intelligent agents: theory and practice" knowledge Engineering review, January 1995.

[Yao et Liu, 1997] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Network*s, 8(3): 694– 713, 1997.

[Yao, 1999] Yao. Evolving artificial neural networks. PIEEE: Proceedings of the IEEE, 87, 1999.

[Yin et Germay,*1993*] Yin X. and N. Germay (*1993*) : ``*A Fast Genetic Algorithm with sharing scheme using cluster analysis methods in Multimodal function optimization*", in Proceedings of the Artificial neural Nets and Genetic algorithms.