REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DE GUELMA
FACULTE DES SCIENCES ET DE L'INGENIERIE

Ecole Doctorale Nationale
Science et Technologie de l'Information et de la Connaissance

Mémoire de Magister

Présenté au département d'informatique

Spécialité : **Informatique**
Option : **SIC**

Par : Melle **Ouarda ZEDADRA**

Résolution de problèmes de coordination d'agents réactifs dans un environnement incertain

---

JURY

Président	M.C.	KHOULADI KHEIREDDINE	Université Mentouri
Rapporteur	Pr.	SERIDI HAMID	Université de Guelma
Examinateur	M.C.	CHAOUI ALLOUA	Université Mentouri
Examinateur	Dr.	LAFIFI YACINE	Université de Guelma

2009

Abstract

The objective of this work is to conceive in a automatic way a collective of agents, plunged into the same environment, endowed with partial perceptions and with incomplete knowledge, which from their interactions are going to evolve together to resolve a problem initially unknown.

The agents concerned by this problem are robots charged to collect of the ore, stemming from the relative problem with care of robots explorers. To resolve the problem of interaction and coordination between the collective of agents, we use the formalism Interac-DEC-POMDP for the modelling of the system and the algorithms of learning and resolution [Vincent, 2005] for the resolution of this formalism.

Keywords: reactive Agents, formalism Interac-DEC-POMDP, learning by intensification, robots explorers.

Résumé

L'objectif de ce travail est de concevoir de manière automatique un collectif d'agents, plongés dans un même environnement, dotés de perceptions partielles et de connaissances incomplètes, qui de par leurs interactions vont évoluer ensemble pour résoudre un problème initialement inconnu.

Les agents concernés par ce problème sont des robots chargés de collecter du minerai, issus de la problématique relative à l'application des robots fourrageurs. Pour résoudre le problème d'interaction et de coordination entre le collectif d'agents, nous utilisons le formalisme Interac-DEC-POMDP pour la modélisation du système et les algorithmes d'apprentissage et de résolution [Vincent, 2005] pour la résolution de ce formalisme.

Mots clés : Agents réactifs, formalisme Interac-DEC-POMDP, apprentissage par renforcement, robots fourrageurs.

الخلاصـــــــــة

الهدف من هذا العمل هو التطوير الآلي لمجموعة من الوكلاء (العمال), يتعايشون في نفس البيئة, مزودون برؤية محدودة و معرفة غير تامة, عن طريق تفاعلهم معا يصلون إلى حالة ثابتة تمكنهم من إيجاد حل لمشاكل معقدة غير معروفة مسبقا.

الوكلاء المعنيون ببحثنا هم وكلاء يعملون على جمع أنواع من الخام النادر, هذه المشكلة مستنبطة من المشكلة المشهورة منذ عدة أعوام ألا وهي مشكلة الوكلاء الحمالون. نعتمد في هذا الحل على المخطط Interac-DEC-POMDP, لبناء النظام , و على خوارزميا التعلم و التشغيل لحل هذا المخطط.

الكلمات الرئيسية: وكيل متفاعل, المخطط Interac-DEC-POMDP, تعزيز التعلم, وكيل حمال.

REMERCIEMENTS

En premier lieu, je tiens à exprimer ma profonde gratitude à mon encadreur Monsieur HAMID SERIDI de m'avoir proposé ce sujet et diriger constamment de près mon travail. Je le remercie pour sa disponibilité, ses encouragements, son soutien moral, sa sympathie et son amitié sincère dont il a fait preuve le long de ce travail. J'apprécie la confiance qu'il m'a témoignée et les conseils avisés qu'il m'a prodigué. Qu'il trouve ici mes sincères reconnaissances.

Je tiens aussi à remercier Madame Benhamza Karima pour son aide, ses inestimables conseils, ses explications claires et ses orientations précises. Qu'il trouve ici mes sincères reconnaissances.

Je suis très reconnaissante à Monsieur Khouladi Kheireddine de m'avoir fait l'honneur de présider mon jury, je lui suis très reconnaissante pour l'intérêt qu'il a porté à mon travail.

Merci à monsieur Chaoui Alloua d'avoir accepté de faire partie de mon jury, je la remercie pour l'intérêt qu'il a porté à mon travail.

Un grand merci à Monsieur Lafifi Yacine d'avoir accepté de faire partie de mon jury, je le remercie pour l'intérêt qu'il a porté à mon travail.

Je remercie également tous les membres de l'équipe de Monsieur SERIDI pour leur soutien moral et leur encouragement pour l'accomplissement de ce travail.

Enfin, je remercie en particulier ma famille et plus précisément ma mère et mon père qui m'ont encouragée tout au long de mes études, je n'oublie pas mes amies Samira et Samia, ainsi que mes collègues Ilyes, Samir, Ibrahim et Khaled pour leur amitié et encouragement.

Liste des figures

Liste des tableaux

Liste des Algorithmes

Sommaire

Sommaire

Chapitre II : Les Processus Décisionnels de Markov et leurs extensions

Chapitre III : Apprentissage dans les systèmes multi-agents

Partie B : Mise en oeuvre

Chapitre IV : Conception

Chapitre V : Expérimentation et résultats

Introduction Générale

Introduction générale

L'intelligence est une notion difficile à cerner. Derrière ce terme se cachent de nombreuses acceptions différentes. Un système pourra être qualifié d'intelligent parce que pour un observateur extérieur, il semblera doté de capacités cognitives habituellement attribuées à l'homme, ou parce qu'il cherchera à reproduire les mécanismes par lesquels l'homme ou l'animal prend des décisions complexes.

L'intelligence artificielle est inspirée de la métaphore du penseur solitaire : les chercheurs dans ce domaine ont cherché à produire des programmes isolés, en émulant les processus cognitifs humains pour résoudre des problèmes complexes. Les problèmes sont parfois naturellement posés de manière distribuée ; on suppose qu'il existe un certain nombre d'entités capables d'agir et d'interagir ; ces problèmes sont ainsi inscrits dans une branche de l'IA : intelligence artificielle distribuée (IAD). L'apparition de l'intelligence artificielle distribuée (IAD) a remis en question l'étude des systèmes constitués d'un agent. Elle a pour objectif de réaliser des organisations de systèmes, capables de résoudre des problèmes par le biais d'un raisonnement symbolique. La métaphore du penseur solitaire a été remise en question et s'est accompagnée d'une nouvelle problématique : celle de l'interaction entre plusieurs entités. Une nouvelle question se pose alors : Comment un agent peut-il prendre en considération la présence d'autres acteurs dans le système, pour interagir au mieux avec eux ?

Le domaine dans lequel s'inscrit notre travail est celui des systèmes multi-agents (SMA). L'approche multi-agents concerne l'étude et la conception des systèmes distribués, faisant interagir plusieurs agents. Cette approche se situe au carrefour de nombreuses disciplines les deux plus importantes sont l'*intelligence artificielle distribuée(IAD)* et la *vie artificielle (VA)* qui cherche à comprendre et à modéliser des systèmes doués de vie, c'est-à-dire capable de survivre, de s'adapter et de se reproduire dans un milieu parfois hostile.

À la différence de l'IA et de la plupart des programmes informatiques, les SMA ne sont plus des penseurs renfermés sur leurs propres raisonnements qui ignorent leur environnement, mais constituent une véritable société d'être qui doivent se mouvoir, planifier, communiquer,

percevoir, agir, réagir et d'une manière générale vivre et travailler dans un milieu dans lequel ils entrent parfois en conflit avec d'autres agents. Pour l'IA c'est l'individu qui est intelligent, alors que pour les SMA c'est l'organisation qui présente des fonctionnalités, que l'on peut qualifier d'intelligentes. Concevoir des êtres intelligents pour les SMA n'est pas un but, puisqu'ils s'intéressent essentiellement aux interactions entre individus et aux conséquences de leurs interactions.

On cherche à construire des systèmes composés d'un contrôle décentralisé pouvant améliorer le système à un tout cohérent. Plusieurs démarches sont envisageables pour la construction de tels systèmes ; certains se fondent sur la conception de méthodologies permettant de faciliter le processus d'ingénierie des systèmes par un ensemble de méthodes appliquées tout au long du cycle de développement d'un logiciel, on peut citer par exemple la méthode GAIA, Cassiopé..., mais c'est sous la responsabilité du concepteur à construire les comportements des agents. Une autre démarche est de s'inspirer des systèmes collectifs observés dans la nature comme métaphore pour proposer de nouveaux mécanismes ; ces méthodes sont aussi intéressante, mais nécessite un travail important de la part du concepteur. Une autre approche très intéressante et qui intègre le moins le concepteur dans la boucle de construction du système : Ce sont les approches fondées sur des cadres formels. Ces derniers sont des outils mathématiques permettant la représentation d'un système, son exécution, ainsi que les problèmes à résoudre, en utilisant des éléments syntaxiques, qu'on peut manipuler ou résoudre à travers des algorithmes de résolution qui existent déjà, ou que l'on propose ; plusieurs formalismes ont vu le jour dans le domaine de prise de décision, s'intéressant à la construction automatique des comportements dans des environnements incertains, dans le cas mono-agent comme les PDM et les POMDP et aussi pour le cas multi-agents comme les MMDPs et les DEC-POMDPs. Dans ces derniers, des interactions indirectes passent entre agents via l'environnement, et elle ne permettant pas au système de détecter qui est à l'origine de cette interaction pour le récompenser ou le pénaliser ; pour pallier à ce problème-la, un nouveau formalisme qui est une extension des DEC-POMDP : Interac-DED-POMDP est ainsi proposé [Vincent, 2005] ; c'est ce dernier que nous utiliserons comme source dans notre travail.

Contexte scientifique :

Ce mémoire se situe à la rencontre de deux champs de recherche : les SMA d'une part, qui constituent une sous branche de l'IA, s'intéressant aux comportements intelligents résultant

de l'activité coopérative de plusieurs agents, et la théorie de décision d'autre part, qui est une discipline mathématique s'intéressant à la notion d'incertitude et de préférence. Cette dernière, propose des éléments syntaxiques permettant de formaliser des problèmes de prise de décision dans un environnement incertain, et des outils permettant de calculer les réponses à ces problèmes.

Nous utiliserons les outils combinés de ces domaines à savoir : « agent rationnel et autonome », « processus décisionnels de Markov », « INTERAC-DEC-POMDP », « interaction directe », « apprentissage par renforcement » pour articuler ce mémoire.

Problématique :

On cherche à concevoir un SMA réactif, caractérisé par un contrôle décentralisé, où chaque agent n'a qu'une vision partielle de l'environnement dans le quel il est plongé. Les actions qu'il peut émettre ne peuvent modifier l'environnement que localement ; et chaque agent décide localement des actions à entreprendre, en suivant ses perceptions reçues par ses capteurs ; mais tous les agents contribuent à l'évolution du système. Pour avoir un système cohérent et performant dans sa globalité, les agents doivent favoriser l'interaction qui leur donne plus de récompenses.

Afin de faciliter la conception d'un SMA, on utilise un cadre formel (les processus markoviens) pour exprimer le problème ; il existe déjà des cadres formels permettant de représenter des agents devant résoudre un problème dans un environnement incertain ; mais ces derniers n'incluent pas la notion de sociabilité, pour cette raison Vincent Thomas propose [Vincent, 2005] un nouveau formalisme permettant de représenter un SMA tout en incluant l'existence d'autres agents dans le système, c'est-à-dire que le comportement de l'agent va prendre en compte la présence des autres agents dans le système. Dans notre conception, nous utiliserons le formalisme INTERAC-DEC-POMDP pour exprimer le système, et un algorithme d'apprentissage [Vincent, 2005] pour construire les comportements des agents.

Plan de mémoire :

Ce manuscrit se décompose en trois parties principales : la première partie constitue une partie d'état de l'art ; elle contient les différentes notions utilisées dans notre travail, ainsi qu'une étude détaillée tant sur les formalismes markoviens et leurs extensions (Interac-DEC-POMDP), que sur les techniques d'apprentissage en univers multi-agents et des techniques d'apprentissage par renforcement ; la deuxième partie porte sur le modèle de conception et sur une définition de l'application de simulation utilisée(les robots fourrageurs), ainsi que sur la

réalisation de la conception et les résultats obtenus. Enfin, la dernière partie contient un résumé du travail fait, et présente les perspectives ouvertes par notre travail.

Dans la première partie, nous commençons par présenter différentes définitions et caractéristiques des agents, à savoir la sociabilité, la rationalité, l'autonomie, la mobilité et l'adaptabilité ; nous passerons ensuite à l'exposition des différentes caractéristiques des agents et de l'environnement dans lequel ils évoluent ; puis nous expliquerons la notion d'intelligence des agents (cognitifs ou réactifs), ensuite nous définirons un système multi-agents (SMA) ainsi que ses caractéristiques ; après nous mettrons l'accent sur l'interaction permettant le passage des comportements individuels à un comportement collectif tout on exposant les différents types d'interactions et leurs différentes situations ; nous introduisons aussi la notion de coordination et les cases dans lesquelles on l'utilise. Enfin, nous aborderons la problématique de prise de décision dans les SMA et nous déterminerons ensuite, les avantages d'utilisation des SMA, ainsi que leurs défis. Nous terminerons la partie par les domaines d'applications des SMA, puis une conclusion.

Nous décrirons ensuite les cadres formels PDM et leurs extensions, à savoir le formalisme INTERAC-DEC-POMDP présenté comme une extension des cadres formels markoviens DEC-POMDP (mais qui inclut la notion d'interaction directe), utilisés pour représenter les constituants d'un SMA.

Nous exposerons les différentes méthodes d'apprentissage utilisées en univers multi-agents, ainsi que les techniques d'apprentissage par renforcement et ses différents paramètres.

La deuxième partie concerne la conception des agents proposés, puis on expliquera comment représenter les politiques d'action, de déclenchement et de résolution d'interaction associes à chaque agent et leurs apprentissages, nous introduisons aussi l'algorithme d'apprentissage utilisé. Celui-ci est décrit de manière détaillée dans [Vincent, 2005], par la suite nous présenterons l'architecture interne, en déterminant au fur et à mesure l'utilité de chacun des modules de cette architecture, puis on passera à une présentation de l'application des robots fourrageurs utilisés, ainsi que l'architecture interne des agents proposée, on passe en revue sur les simulations et les résultats obtenus.

Enfin, la dernière partie contient un résumé du travail fait et des résultats obtenus, et présente les perspectives à court et à long terme ouvertes par notre travail.

Partie A

De l'Agent aux Systèmes Multi-Agents

Partie A

De l'Agent aux Systèmes Multi-Agents

Introduction

À la différence de l'Intelligence Artificielle (IA) qui modélise le comportement intelligent d'un seul agent, l'intelligence artificielle distribuée (IAD) s'intéresse à des comportements intelligents qui résultent de l'activité coopérative de plusieurs agents. Suite à la distribution de l'expertise sur un ensemble de composants qui communiquent pour atteindre un objectif global, ou résoudre un problème, il est nécessaire de diviser le problème en sous-problèmes. Cette division n'est pas toujours aisée, car beaucoup de problèmes ne peuvent être divisés. Ainsi, une extension des systèmes d'IAD est proposée : les composants doivent être capables de raisonner sur les connaissances et les capacités des autres, dans le but d'une coopération effective. Pour ce faire, ils doivent être dotés de capacités de perception et d'action sur l'environnement, et doivent posséder une certaine autonomie de comportement ; on parle alors d'agents et par conséquent de système multi-agents.

Dans cette partie, nous décrirons les systèmes que nous souhaitons construire au niveau conceptuel. Cette partie a plusieurs objectifs. Tout d'abord, présenter la diversité des définitions du terme agent, tout en décrivant les différentes caractéristiques de ce terme. Nous introduirons ensuite la notion des systèmes multi-agents, en mettant l'accent sur la notion d'interaction fondamentale dans ces systèmes, en passant ensuite à la notion de coordination et les algorithmes mises pour la résolution des problèmes de coordination. Nous reviendrons enfin sur certaines propriétés des systèmes multi-agents, qui occupent une place essentielle dans la suite de notre travail. La seconde partie, traite les processus de décision markoviens et leurs extensions. Nous décrirons les processus décisionnels de Markov décentralisés partiellement observables incluant la notion d'interaction directe (Interac-DEC-POMDP). Dans la troisième partie nous exposerons les techniques d'apprentissage dans le cadre des systèmes multi-agents, et on aura recours à l'apprentissage par renforcement où on explique ses différents algorithmes.

Chapitre I

Les Systèmes Multi-Agents

Chapitre I

Les Systèmes Multi-Agents

1.1Introduction :

Nous commencerons ce chapitre par la présentation des différentes définitions du terme agent, en passant en revue sur les propriétés d'un agent à savoir la rationalité, l'autonomie, la réactivité, l'adaptabilité, la mobilité et la sociabilité, les caractéristiques d'un agent et de son environnement, aussi on donnera les deux types d'agents (cognitifs ou réactifs), et on parle de la mémoire d'un agent réactif.

Nous procéderons ensuite, à introduire les différentes notions concernant un système multi-agents, ses caractéristiques et ses typologies.

Nous exposerons par la suite, la notion d'interaction qui constitue l'essence des SMA et nous définissons ses situations d'interaction, puis on définira la notion d'interaction directe et indirecte. On passe à définir les deux structures organisationnelles pour une société d'agents.

Nous passerons à définir la coordination d'action et les problèmes qui on découle ainsi que les formes de coordination existantes.

On passe en revue sur la problématique de prise de décision dans un SMA et nous procéderons à l'exposition des avantages et les défis des SMA ainsi que leurs domaines d'application.

1.2 Présentation du concept Agent :

Le concept agent a été l'objet d'études pour plusieurs décennies, dans différentes disciplines. Il a été non seulement utilisé dans les systèmes à base de connaissances, la robotique, le langage naturel et d'autres domaines de l'intelligence artificielle, mais aussi dans des disciplines comme la philosophie et la psychologie. Aujourd'hui, avec l'avènement de nouvelles technologies et l'expansion d'internet, ce concept est encore associé à plusieurs nouvelles applications, comme agent ressource, agent courtier, assistant personnel, agent interface, agent ontologique, etc. Dans la littérature, on trouve une multitude de définitions

d'agents. Dans ce qui suit, nous présenterons quelques définitions plus importantes.

1.2.1 Définitions :

Définition 1 : *Définition du mot agent selon le dictionnaire Larousse*

" Celui qui agit, produit un effet " est la définition de l'agent selon le dictionnaire Larousse de la langue française. Cette définition, même si elle correspond partiellement au sens des systèmes multi-agents, reste cependant incomplète.

Définition 2 : *Agent selon Russell et Norvig*

L'une des définitions les plus célèbres de la notion d'agent a été formulée par [Russell et Norvig, 1995] ; ils considèrent un agent comme *" Tout ce qui peut être vu comme percevant son environnement à l'aide de capteurs, et agissant sur cet environnement à l'aide d'effecteurs, de façon autonome".*

Définition 3 : *Agent selon Wooldridge et Jenings*

Selon Wooldridge et Jenings [Wooldridge et Jenings, 1995], *"An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives"*[1]. Par perception de son environnement, un agent peut donc collecter des informations qu'il utilisera ensuite, afin de décider de façon autonome comment agir.

Définition 4 : *Agent selon Jacques Ferber*

Les définitions précédentes du terme agent sont très générales et volontairement minimalistes dans sa formulation, elles ont été étendues notamment par Ferber pour accentuer l'importance de l'environnement. Pour [Ferber, 1995] :

« L'agent est une entité physique ou virtuelle :

1. *Qui est capable d'agir dans un environnement,*
2. *Qui peut communiquer directement avec d'autres agents,*
3. *Qui est mue par un ensemble de tendances (sous la forme d'objectifs individuels ou de fonctions de satisfaction, voir de survie, qu'elle cherche à optimiser),*
4. *Qui possède des ressources propres,*
5. *Qui est capable de percevoir son environnement (mais de manière limitée),*

[1] Un agent est un système informatique situé dans un environnement et capable d'agir de manière autonome dans cet environnement afin d'atteindre ses objectifs.

6. *Qui ne dispose que d'une représentation partielle de cet environnement (ou éventuellement aucune),*
7. *Qui possède des compétences et offre des services,*
8. *Qui peut éventuellement se reproduire,*
9. *Dont le comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont elle dispose, et en fonction de sa perception, de ses représentations, et des communications qu'elle reçoit. »*

Définition 5 : *Agent selon Jennings et al*

Jennings, Sycara et Wooldridge [Jennings et al, 1998] ont proposé la définition suivante pour un agent: *« Un agent est un système informatique, situé dans un environnement, et qui agit d'une façon autonome et flexible, pour atteindre les objectifs pour lesquels il a été conçu ».*
Les notions situées, autonome et flexible sont définies comme suit [Chaib-draa, 1999]:

- *Situé:* l'agent est capable d'agir sur son environnement à partir des entrées sensorielles qu'il reçoit de ce même environnement. Exemples: systèmes de contrôle de processus, systèmes embarqués, etc. ;
- *Autonome:* l'agent est capable d'agir sans l'intervention d'un tiers (humain ou agent) et contrôle ses propres actions ainsi que son état interne;
- *flexible:* l'agent dans ce cas est:
  - ✓ *Capable de répondre à temps:* l'agent doit être capable de percevoir son environnement et élaborer une réponse dans les temps requis,
  - ✓ *Proactifs :* l'agent doit exhiber un comportement proactif et opportuniste, tout en étant capable de prendre l'initiative au bon moment;
  - ✓ *Social:* l'agent doit être capable d'interagir avec les autres agents (logiciels et humains), quand la situation l'exige, afin de compléter ses tâches ou aider ces agents à accomplir les leurs.

Discussion sur les différentes définitions :

En comparant les définitions ci-dessus, nous pouvons identifier deux tendances principales, quand nous définissions les agents. Quelques chercheurs considèrent que nous pouvons parler et définir un agent en isolation, alors que certains d'autres considèrent les agents principalement comme entités agissant dans une société avec d'autres agents, donc le paradigme des systèmes multi-agents (SMA). Les deux tendances ont déjà porté des résultats. C'est vrai de penser que c'est le paradigme des SMA qui va s'imposer comme prépondérant, car il est plutôt difficile de compter qu'un agent existe seulement comme une entité pour lui

seul, et ne rencontrera pas d'autres agents (soit artificiels ou humains) dans son environnement. Les agents personnels, ou les agents d'information, qui ne sont pas censés principalement de travailler collectivement pour résoudre des problèmes, auront certainement beaucoup à gagner s'ils agissent par interaction avec d'autres agents.

1.2.2 Les propriétés d'un Agent :

Les propriétés qui seront expliquées dans ce qui suit, influencent essentiellement le comportement d'un agent, et ce sont elles qui vont conditionner la conception de cet agent.

Rationalité :

La vision la plus simple qu'on peut utiliser pour décrire un agent, est celle d'une entité qui perçoit son environnement par des détecteurs, et agit sur l'environnement à travers des effecteurs. On aimerait qu'un agent intelligent exécute des tâches pour nous et que, en même temps, il ait un comportement rationnel.

Définition 6 : ***Agent rationnel selon Russell et Norvig [Russell et Norvig, 1995]***

« Un agent rationnel est un agent qui agit d'une manière lui permettant d'obtenir le plus de succès possible dans la réalisation des tâches qu'on lui a assignées ».

Définition 7 : ***Rationalité selon Herbert Simon [Simon, 1947]***

« Il est impossible que le comportement d'un seul individu isolé atteigne un degré de rationalité élevé. Le nombre d'alternatives à explorer est si grand, l'information nécessaire à leur évaluation si vaste que même une approximation de la rationalité objective est difficile à concevoir »

Simon [Simon, 1947] a mis en évidence l'importance majeure des perceptions et des connaissances dans la prise de décision. Il a également étudié les limites de la rationalité des décideurs, et a introduit la distinction entre la rationalité optimisante[2], et la rationalité limitée[3].

Comment peut-on mesurer le succès d'un agent rationnel? Pour cela, il faut disposer d'une mesure de performance, si possible objective, associée à une certaine tâche que l'agent doit exécuter. La mesure de performance doit être rapportée aux capacités de l'agent[Russell et Norvig, 1995], notamment à ce qu'il a perçu sur l'environnement pendant le temps à

[2] Appelée aussi parfaite ou objective, supposait que le décideur disposait d'une connaissance parfaite de toute ses possibilités, et des conséquences de ses décisions ; son choix sera donc optimal.

[3] Si les capacités des décideurs sont restreintes et les éléments pertinents pour la prise de décision sont incomplètes, il ya recours donc a la recherche d'une solution satisfaisante, et donc on parle de rationalité limitée.

considérer (sa séquence de perception), à ce qu'il connaît sur son environnement, et aux actions qu'il peut effectuer. La rationalité de l'agent construit peut alors être évaluée en tenant compte de tous ces aspects. Pour prendre en compte ces différents facteurs, Russell et Norvig [Russell et Norvig, 2003] définissent la notion d'agent rationnel comme suit :

Définition 8 : *Agent rationnel selon Russell et Norvig[Russell et Norvig, 2003]*

« Pour chaque séquence possible de perceptions, un agent rationnel devrait sélectionner l'action pour laquelle il s'attend à maximiser sa mesure de performance, étant données les preuves fournies par sa séquence de perception et quelles que soit ses connaissances à priori ».

Autonomie:

Un agent autonome est un agent qui agit dans et sur son environnement et qui ne se contente pas d'être dirigé par des commandes venant d'un autre agent, ou de l'opérateur. Un tel agent doit posséder des effecteurs, et être situé dans son environnement. L'autonomie d'un agent ne porte pas seulement sur son comportement, mais aussi sur ses ressources internes qui sont elles aussi autonomes ; ainsi il pourrait être intéressant d'avoir des programmes qui prennent l'initiative dans certains circonstances ; ainsi un logiciel prendrait maintenant une part active pour aider l'usager à accomplir ses tâches.

Pour Jose Brustoloni [Brustoloni, 1991] un agent autonome doit savoir comment accomplir ses buts en fonction de l'état de son environnement, ou du moins, doit être capable de trouver un moyen pour accomplir ses buts. Ces capacités de connaissance ou de recherche, impliquent un traitement structurel (c'est le cas des architectures connectivites) ou symbolique (où l'agent manipule des symboles et détermine son comportement en fonction de ces manipulations) des stimuli reçus.

Définition 9 : *Agent autonome selon Russel et Norvig[Russell et Norvig, 2003]*

Un système autonome est défini comme suit : *"Un système est autonome dans le sens ou son comportement est déterminé par ses propres expériences".*

Un agent, selon cette définition est un agent qui parvient à adapter son comportement en utilisant ses expériences passées. Un tel agent possède alors de grandes capacités d'adaptation à des environnements inconnus. Nous adopterons cette définition dans notre travail.

Réactivité:

La réactivité, c'est la capacité d'un agent à percevoir son environnement et à décider

comment agir, en respectant le temps qui lui est imparti. Un tel agent est en mesure de prendre ses décisions en temps réel. Les agents réactifs basent leurs comportements sur un ensemble de stimulus-réponse, chaque stimulus (ou situation) fait correspondre une réponse (ou action). Tout agent reçoit un ensemble de perceptions qui le conduit à exécuter des actions ; le passage des perceptions aux actions est réalisé par un processus délibératif, qui permet à l'agent de décider quelle action exécuter. De manière générale, plus un mécanisme délibératif sera simple, plus l'agent fera preuve d'une grande réactivité.

Adaptabilité:

Un agent adaptatif est capable d'apprendre en fonction de son expérience passée et de son évolution. Les architectures d'agents produisant des unités fonctionnelles dédiées à des tâches précises, les techniques d'apprentissage classique de l'intelligence artificielle s'appliquent particulièrement bien à l'apprentissage d'un agent (qu'il s'agisse de faire évoluer ses capacités d'action, de décision ou d'analyse) [Sigaud, 2004].

La mobilité:

Un agent mobile, est un agent capable de se déplacer dans son environnement, qui peut être physique réel ou simulé. Un agent mobile dispose donc de dispositifs assurant sa mobilité. Dans le cas d'un agent robotique physique ou simulé, il s'agit bien évidemment d'effecteurs lui permettant de se déplacer. Dans le cas d'un agent logiciel, la mobilité implique le déplacement dans le réseau ou dans l'architecture du système, c'est-à-dire la mise en place d'un mécanisme de migration de processus. Un agent robotique mobile peut se déplacer vers les ressources dont il a besoin, se rapprocher d'un objectif à accomplir, fuir des prédateurs, etc. Un agent logiciel mobile n'est bien entendu pas concerné par la problématique de la localisation physique, mais en revanche, il peut tirer parti de la topologie du réseau informatique [Gray, 1998], [Rus, 1996].

Les agents d'information sont la plupart du temps des agents mobiles: leur rôle est de parcourir des sources d'information souvent distribuées et donc d'être amenés à se déplacer dans un réseau. Des agents d'information mobiles permettent à l'utilisateur de s'abstraire de la nature des sources d'information, à condition que l'agent mobile assure un rôle de passerelle et soit pourvu d'une interface avec ces sources d'information. De même, grâce à ses ressources et à ses buts propres, un agent mobile peut effectuer sa recherche d'informations de manière asynchrone sans que l'utilisateur n'ait à le manipuler [Rus, 1996].

La sociabilité:

Un agent social est un agent qui opère des interactions avec d'autres agents, au sein de son environnement. Ces interactions peuvent faciliter la tâche d'un agent (coopération) ou au contraire le gêner dans l'accomplissement de ses buts (encombrement, compétition). Un système possédant plusieurs agents est appelé un système multi-agents ou SMA. Nous appellerons un système possédant plusieurs agents sociaux, un système multi-agents social.

Les agents avec lesquels un agent social est en mesure d'avoir des interactions, peuvent être hétérogènes, c'est-à-dire qu'ils peuvent avoir des formes, des interfaces, des buts ou des représentations de l'environnement différent. La communication est une forme d'interaction nécessitant un protocole et un langage commun aux différentes parties, et indépendant des spécificités de ces dernières. De plus, un canal de communication adapté à tous les agents l'utilisant, et permettant de transporter les messages échangés et requis [Bousquet et al, 2001]. Les interactions et plus spécifiquement, la communication sont des composantes nécessaires au travail coordonné d'une équipe d'agents.

Discussion :

Dans la section précédente nous avons présenté les différentes propriétés que peut avoir un agent ; les agents dont nous voulons concevoir, sont des agents qui n'ont pas de représentation de leurs environnements, mais ont des perceptions partielles ; donc ils sont des agents réactifs, ils doivent agir sans intervention d'un tiers, donc ils sont autonomes, les agents ensembles doivent avoir un comportement global du système cohérent et performant ; pour cela ils doivent être rationnels, ils ont la possibilité d'interagir directement entre eux par envoi de signaux, donc ils sont sociables et peuvent se mouvoir d'un emplacement à l'autre ; alors ils sont des agents mobiles (robotique). Dans ce qui suit nous expliquerons quelque propriété qui devront être présentes dans notre conception.

1.2.3 Les caractéristiques d'un Agent :

Afin que l'agent puisse agir rationnellement au sens défini précédemment le concepteur doit, décrire pour cet agent [Vincent, 2005] :

- Les capacités de perception et d'action sur l'environnement dans lequel il est plongé.
- Les représentations internes, c'est-à-dire l'ensemble des structures de données et des processus internes à un agent lui permettant de prendre une décision.

La suite permettra de présenter ces différentes facettes.

Localité des perceptions et d'actions :

Chaque agent est situé et est limité dans ses capacités d'action et de perception. Les agents ne perçoivent et ne peuvent modifier l'environnement que d'une façon partielle (et donc locale).

Caractéristiques de l'environnement :

Un agent est situé dans un environnement. Pour modéliser la structure de l'agent, il faut avoir un modèle de l'environnement. Ce dernier, peut être vu comme étant dans un état e parmi un ensemble d'états E= $\{c_1, .., c_n, ..\}$, peut changer son état soit d'une manière spontanée soit comme résultat des actions de l'agent. L'évolution de l'environnement se modélise différemment selon les caractéristiques que l'on prend en compte, et les simplifications que l'on s'autorise. Les principales distinctions à faire sur les types d'environnements sont [Russell et Norvig, 2003]:

– *Environnement accessible ou inaccessible :* Un environnement est accessible à l'agent si celui-ci peut percevoir entièrement l'état de l'environnement ou au moins tous les traits de l'environnement qui sont significatifs du point de vue des actions de l'agent. Sinon, l'environnement est inaccessible, on parlera alors d'environnement totalement observable ou partiellement observable.

– *Environnement déterministe ou non déterministe :* Si l'état suivant de l'environnement est déterminé d'une manière unique par l'état courant et l'action de l'agent, alors l'environnement est déterministe. Si le résultat est incertain, notamment si, par suite d'une action de l'agent, l'environnement peut évoluer de différentes manières, alors on est dans le cas non déterministe ou encore stochastique.

–*Environnement statique ou dynamique :* Si l'environnement ne peut pas changer d'état sans l'intervention de l'agent, on est dans le cas statique. L'environnement est dynamique si son état peut se modifier sans l'action de l'agent dans l'intervalle de temps entre deux perceptions de l'agent.

–*Environnement discret ou continu :* Si tout passage d'un état de l'environnement à un autre nécessite le passage par une séquence d'états intermédiaires, alors on a un environnement continu ; autrement, l'environnement est discret.

–*Environnement épisodique ou séquentiel :* dans un environnement épisodique, l'expérience de l'agent peut être divisée en épisodes, chaque épisode consiste en une phase de perception, suivie de l'exécution d'une action ; chaque épisode est indépendante des autres et le résultat de l'exécution d'action ne dépend que de l'épisode courante. Dans un

environnement séquentiel, l'exécution d'une action peut influencer toutes les actions à venir [Beynier, 2006].

Les caractéristiques de l'environnement influencent la façon dont on conçoit un agent, car il faut tenir compte de l'évolution de l'environnement, de la capacité de l'agent de saisir cette évolution, et de sa capacité à décider en conséquence. Par exemple, si on a plusieurs agents qui agissent dans un même environnement, chaque agent va percevoir l'environnement comme dynamique et non déterministe, vu que l'état de l'environnement changera en raison des actions des autres agents, et une même action exécutée dans un certain état aura des résultats différents en fonction des actions de ces autres agents.

Selon [Hewitt, 1986], les environnements suivants partiellement observables, stochastiques, séquentiels, dynamiques et continus, sont classés comme des environnements ouverts, d'une grande complexité. Wooldridge [Wooldridge, 2002] argumente que la qualité de la décision dépend de la qualité de l'information disponible ; plus l'agent aura d'informations fiables pour prendre une décision, plus celle-ci s'avérera adaptée à la situation.

Intelligence des Agents :

Faut-il concevoir les agents comme des entités déjà *intelligentes* c'est-à-dire capables de résoudre certains problèmes par eux-mêmes, ou bien faut-il les assimiler à des êtres très simples réagissant directement aux modifications de l'environnement ? [Ferber, 1995]. Selon cette question on peut déduire qu'à partir de la représentation interne des agents on peut dégager deux types d'agents qui sont :

a) Agents cognitifs :

Pour appliquer le modèle de l'intelligence humaine et la perspective humaine sur le monde, il est courant dans la communauté des chercheurs d'intelligence artificielle de caractériser un agent intelligent en utilisant des notions mentales telles que la connaissance, la croyance, les intentions, les désirs, les choix et les engagements [Shoham, 1993]. Une des caractéristiques les plus importantes des agents intelligents est qu'ils peuvent être vus en tant que systèmes intentionnels [Dennett, 1987]. Comme le précise Shoham [Shoham, 1993], une perspective mentale ou intentionnelle des agents intelligents n'est pas simplement une autre invention des informaticiens, mais un paradigme utile pour décrire les systèmes répartis complexes. La complexité d'un tel système, dont nous ne pouvons pas connaître ou prévoir la structure interne de tous les composants, semble impliquer que nous devons compter sur une explication intentionnelle du fonctionnant du système.

De tels agents intelligents, principalement caractérisés par un niveau symbolique de la représentation des connaissances, et par des notions mentales, sont nommés d'habitude des agents cognitifs. Ils possèdent une représentation complète de l'environnement, des buts explicites, capables de planifier leur comportement, mémoriser leurs actions passées, communiquer par envoi de messages, négocier, etc. Un SMA constitué d'agents cognitifs possède communément peu d'agents [Georgeff et al, 1999].

b) Agents réactifs

Un agent est réactif s'il répond de manière opportune aux changements de son environnement. Ceux-ci peuvent être constitués de stimuli externes, ils sont souvent qualifiés de ne pas être *intelligents* par eux-mêmes. Ils sont des composantes très simples qui perçoivent l'environnement, et sont capables d'agir sur celui-ci. Ils n'ont pas une représentation symbolique de l'environnement ou des connaissances et ils ne possèdent pas de croyances, pas de mécanismes d'envoi de messages. Leurs capacités répondent uniquement au mode stimulus/action, qui peut être considéré comme une forme de communication. Un SMA constitué d'agents réactifs possède généralement un grand nombre d'agents, et présente un comportement global intelligent [Ferber, 1995]. La structure des agents purement réactifs tend à la simplicité, mais ces derniers peuvent être capables d'actions de groupe complexes et coordonnées. C'est le cas d'une société de fourmis, dont la somme des membres est capable d'actions évoluées, mais dont chaque individu pris séparément possède une représentation faible de l'environnement et n'a pas de buts globaux. On trouve un exemple de définition d'une société d'agents réactifs dans [Drogoul et al, 1995][Drougoul et Ferber, 1992] Une simulation (MANTA) met en évidence l'émergence de stratégies à long terme complexes au sein d'une société d'agents pourtant non cognitifs. Cette émergence est aussi mise en exergue dans une simulation de jeu d'échecs, où l'approche distribuée purement réactive produit cependant un comportement consistant sur le long terme.

Comme il a été précisé en introduction, notre objectif est de construire un système multi agent réactif avec cependant un intérêt particulier au deuxième type d'agents.

Mémoire d'un Agent :

Les agents que nous souhaitons concevoir doivent être autonomes et rationnels, c'est-à-dire que leurs comportements doivent être déterminé et modifié par leurs expériences. Pour pouvoir garder une trace de leurs expériences et adapter leurs comportements en conséquence, un agent a donc besoin de mémoire.

Pour un agent réactif, on distinguera deux types de mémoire [Vincent, 2005]

- *La mémoire à court terme* . qui est destinée à stocker des événements précis, afin de modifier les décisions de l'agent en fonction d'une (ou plusieurs) observation (s) passée(s). Ce type de mémoire intervient directement comme entrée des règles comportementales stimulus-réponse de l'agent. Une question reste cependant posée : celle consistant à déterminer quel élément peut être utile à mémoriser.
- *La mémoire à long terme :* qui a pour objectif d'adapter le comportement de l'agent. Cette mémoire est constituée d'une synthèse de l'ensemble de l'expérience de l'agent. Ce type de mémoire n'est pas une entrée directe des règles comportementales, mais un moyen de remettre en cause les règles stimulus-réponse internes à l'agent, et d'en produire éventuellement de nouvelles. La question posée dans ce cadre, consiste à trouver comment faire la synthèse des expériences passées, comment la stocker et comment adapter le comportement de l'agent à partir de cette synthèse ?

1.3 Les systèmes multi-agents :

Le domaine des systèmes multi-agents (SMA) est actuellement un domaine de recherche, qui suscite beaucoup d'intérêt. Ce domaine est né à la fin des années 70 et début des années 80, de l'idée de distribuer les connaissances et le contrôle dans les systèmes d'intelligence artificielle. Cette idée a émergé d'une part, du besoin de faire face à la complexité croissante de ces systèmes, et a été favorisée d'autre part par l'émergence des modèles et machines parallèles, rendant possible la mise en œuvre opérationnelle de la distribution [Weiss, 1999].

Les SMA sont à l'intersection de plusieurs domaines scientifiques: informatique répartie et génie logiciel, intelligence artificielle, vie artificielle. Ils s'inspirent également d'études issues d'autres disciplines connexes, notamment la sociologie, la psychologie sociale, les sciences cognitives et bien d'autres. C'est ainsi qu'on les trouve parfois à la base:

- Des systèmes distribués [Fagin et al, 1995].
- D'interface personnes-machines [Kobsa, 1989], [Negroponte, 1995].
- Des bases de données et bases de connaissances distribuées coopératives [Babin et al, 1997].
- Des systèmes pour la compréhension du langage naturel [Allen et al, 1980], [Cohen, 1990a], [Cohen, 1990b], [Kaplan, 1998].
- Des protocoles de communication et réseaux de télécommunications [Fagin et al, 1995], [Malville et Bourdon, 1998], [Nwana et Ndumu, 1999];

- De la programmation orientée agents et génie logiciel [Shoham, 1993], [Thomas, 1993];
- De la robotique cognitive et coopération entre robots [Lukemeyer et Levesque, 1999], [Lespérance, 1991], [Lespérance et al, 1994];
- Des applications distribuées comme le web, l'Internet, le contrôle de trafic routier, le contrôle aérien, les réseaux d'énergie, etc. [Maudet et Chaib-draa, 1996], [Jennings, 1998].

Une bonne description de l'évolution historique de ce domaine est donnée par J. Ferber [Ferber, 1995].

1.3.1 Définition d'un SMA :

Un système multi-agent est un système distribué, composé d'un ensemble d'agents. Contrairement aux systèmes d'IA, qui simulent dans une certaine mesure les capacités du raisonnement humain, les SMA sont conçus et implantés idéalement comme un ensemble d'agents interagissant, le plus souvent, selon des modes de coopération, de concurrence ou de coexistence [Chaib-draa et Levesque, 1994a], [Chaib-draa, 1994b], [Maudet et Chaib-draa, 1996], [Moulin et Chaib-draa, 1996].

Comme pour la définition d'un agent, il existe aussi pour les SMA une multitude de définitions, mais ils sont tous en consensus qu'un système multi-agents est une multiplicité des entités en interaction.

Définition 10 : ***un système multi-agents selon Ferber [Ferber, 1995]***

« Un système multi-agents est un système composé des éléments suivants :

- *Un environnement E, c'est-à-dire un espace disposant généralement d'une métrique.*
- *Un ensemble d'objets O. Ces objets sont situés, c'est-à-dire que pour tout objet, il est possible, à un moment donné, d'associer une position dans E. Ces objets sont passifs, c'est-à-dire qu'ils peuvent être perçus, créés, détruits et modifiés par les agents.*
- *Un ensemble A d'agents qui sont des objets particuliers ($A \subseteq O$), représentant les entités actives du système.*
- *Un ensemble de relations R qui unissent des objets (et donc des agents) entre eux.*
- *Un ensemble d'opérations Op permettant aux agents de A de percevoir, produire, consommer, transformer, et manipuler des objets de O.*
- *Des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers. »*

Définition 11: ***un système multi-agents selon Wooldridge [Wooldridge, 2002]***

« A multi agent system is one that consists of a number of agents, which interact with one another»[4]

Un SMA est une société organisée d'agents dans laquelle un certain nombre de phénomènes peuvent émerger, comme la résultante des interactions entre les agents. L'émergence est essentielle dans les SMA, car c'est l'une des propriétés qui les rendent si aptes à modéliser les systèmes complexes.

1.3.2 Caractéristiques d'un SMA :

L'analyse ou le développement d'un SMA, nécessitent la prise d'au moins quatre dimensions, comme identifiée dans l'approche voyelle AEIO définie par Y. Demazeau [Demazeau, 1995], l'Agent, l'Environnement, l'Interaction et l'Organisation. Selon cette approche, un système multi-agents est constitué par:

- *Des agents (A)* autonomes, chacun étant doté de ses propres senseurs et effecteurs
  D'environnement (E) qui correspond à l'ensemble des éléments décrivant le monde dans lequel évoluent les agents
- *Des interactions (I)* qui correspondent aux couplages qui peuvent s'exercer entre les agents du fait d'actions réciproques et à la manière dont ceux-ci s'exercent.
- *D'organisation (O)* qui peut être perçue comme la résultante globale des comportements des agents.

Un SMA est généralement caractérisé par [Jenings et al, 1998] :

- Chaque agent a des informations incomplètes, ou des compétences limitées pour résoudre le problème ; de ce fait chaque agent a un point de vue limité.
- Il n'existe pas de contrôle centralisé.
- Les données sont décentralisées.

1.3.3 Typologies des SMA :

De la même manière qu'il existe deux types principaux d'agents on les distingue en fonction du type d'agents qui les composent. On compte ainsi deux grandes classes de systèmes multi-agents :

[4] Un SMA est constitué d'agents interagissant entre eux.

Les systèmes multi-agents cognitifs :

Dans le cadre du raisonnement pratique, un raisonnement orienté vers la prise en compte des états mentaux, les chercheurs ont développé l'architecture BDI [Bratman, 1987], [Georgeff et Lansky, 1987], [Rao et Georgeff, 1991], [Singh, 1994](de l'anglais Belief, Desire, Intention pour croyance, désir, et intention), une architecture bâtie autour du raisonnement pratique ; ces agents sont généralement représentés par un état *mental* ayant les attitudes mentales suivantes [Florea et al, 2002]:

–*Les croyances:* Ce que l'agent connaît de son environnement.

–*Les désirs:* Les états possibles envers lesquels l'agent peut vouloir s'engager.

–*Les intentions:* Les états envers lesquels l'agent s'est engagé, et envers lesquels il a engagé des ressources.

Un agent BDI doit donc mettre à jour ses croyances avec les informations qui lui proviennent de son environnement, décidé quelles options lui sont offertes, filtrer ces options afin de déterminer de nouvelles intentions, et poser ses actions au vu de ses intentions. [Rao et George, 1995]

Avantages : De tels systèmes peuvent mettre en œuvre des mécanismes complexes de représentation des autres, de négociation et d'échanges d'information et en tirer partit. Le concepteur peut alors s'inspirer des comportements humains pour guider la construction de système [Vincent, 2005].

Inconvénients : Cette approche se heurte au problème de la représentation des connaissances, problème déjà présent pour des agents isolés. En outre, la représentation complexe des autres agents du système limite le nombre d'agents, et peut mener à des raisonnements coûteux. À cela s'ajoute enfin la complexité des processus de communications entre agents [Vincent, 2005].

Les systèmes multi-agents réactifs :

Un des grands noms parmi les critiques du raisonnement symbolique fut Brooks, qui, par le biais de plusieurs papiers, exposa son opposition au modèle symbolique, et proposa une approche alternative qu'on appelle aujourd'hui IA réactive (SMA réactif). Selon lui, le comportement intelligent devrait émerger de l'interaction entre divers comportements plus simples. Ainsi, au sein de son programme de recherche, il a développé l'architecture subsomption. Dans cette architecture, on bâtit des agents sans utiliser de représentation symbolique, ni de raisonnement. Un agent est alors vu comme un ensemble de comportements

accomplissant une tâche donnée. Chaque comportement est une machine à états finis, qui établit une relation entre une entrée sensorielle et une action en sortie [Briot et Demazeau, 2001].

Typiquement, l'ensemble des comportements est représenté sous forme d'une hiérarchie, dans laquelle les couches de niveau inférieur représentent des comportements moins abstraits, et les couches de niveau supérieur, des comportements plus abstraits. Le développement d'un agent devient donc un processus qu'on devra expérimenter avec les nouveaux comportements. Ceci est habituellement accompli en plaçant l'agent dans son environnement et en observant les résultats.

Avantages : les systèmes multi-agents réactifs présentent habituellement des intérêts en termes de fiabilité (assurée par le grand nombre d'agents du système et leur simplicité), et du passage à l'échelle (adaptation à l'augmentation du nombre d'agents).

Inconvénients : la création de tels systèmes doit faire face aux difficultés de prédiction du comportement global émergent, non représenté explicitement dans le système et à la difficulté de contrôler les comportements individuels par rapport à un objectif non représenté explicitement. Ces difficultés proviennent du fait que la tâche à résoudre et les moyens pour résoudre cette tâche sont exprimés à deux niveaux de complexité distincts. [Vincent, 2005].

1.4 Interaction dans les systèmes multi-agents :

Une des principales propriétés de l'agent dans un SMA, est celle d'interagir avec les autres agents. Ces interactions sont généralement définies comme toute forme d'action exécutée au sein du système d'agents, et qui a pour effet de modifier le comportement d'un autre agent. Elles permettent aux agents de participer à la satisfaction d'un but global. Cette participation permet au système d'évoluer vers un de ses objectifs, et d'avoir un comportement intelligent indépendamment du degré de complexité des agents qui le composent.

1.4.1 Définition :

Définition 12: Interaction selon Ferber [Ferber, 1995]

« Une interaction est une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques... Les interactions sont non seulement la conséquence d'actions effectuées par plusieurs agents en même temps, mais aussi l'élément nécessaire à la construction d'organisation sociale. »

Les interactions entre les agents vivants et leur environnement ont pour but principal de

maintenir la viabilité des agents, en assurant la subsistance et l'élimination des déchets. Les agents réactifs interagissent de manière passive, parce qu'ils entrent en conflit pour des ressources, ou parce qu'ils modifient le monde en laissant des traces dans leur environnement. Les interactions entre les agents ont lieu soit, parce qu'il y a conflit, soit parce qu'il y a coopération pour réaliser une tâche commune. Les agents cognitifs ont des interactions plus variées, en particulier ils ont des interactions intentionnelles et utilisent des systèmes de signes pour communiquer.

1.4.2 Situations d'interaction :

Définition 13: Situation d'interaction selon Ferber [Ferber, 1995]

Selon Ferber c'est : *« Un ensemble de comportement résultant du regroupement d'agents qui doivent agir pour satisfaire leurs objectifs, en tenant compte des contraintes provenant des ressources plus au moins limitées dont ils disposent et de leurs compétences individuelles. »*

Plusieurs types d'interaction ont été définis et analysés à travers diverses composantes [Ferber, 1995]. Ferber donne une classification des situations d'interaction abordée selon le point de vue d'un observateur extérieur. Cette classification présente les différentes situations d'interaction que l'on retrouve en fonction des objectifs des agents (compatibles ou incompatibles), des ressources dont ils disposent (suffisantes ou insuffisantes), et de leurs compétences pour la résolution d'un problème.

Les différentes situations d'interaction sont présentées dans le tableau 1.1 suivant :

Buts	Ressources	Compétences	Situation
Compatibles	Suffisantes	Suffisantes	Indépendance
Compatibles	Suffisantes	Insuffisantes	Collaboration simple
Compatibles	Insuffisantes	Suffisantes	Encombrement
Compatibles	Insuffisantes	Insuffisantes	Collaboration coordonnée
Incompatibles	Suffisantes	Suffisantes	Compétition individuelle pure
Incompatibles	Suffisantes	Insuffisantes	Compétition collective pure
Incompatibles	Insuffisantes	Suffisantes	Conflits individuels pour des ressources
Incompatibles	Insuffisantes	Insuffisantes	Conflits collectifs pour des ressources

Tableau 1.1 : Différentes situations d'interaction [Ferber, 1995]

La collaboration consiste à travailler à plusieurs sur un même projet ; elle désigne les techniques permettant aux agents de se répartir les tâches, les informations et les ressources. Ce concept est aussi différent du concept de coordination d'action, qui est définie comme l'ensemble d'actions supplémentaires que doit accomplir un agent pour améliorer le fonctionnement du système ; elle sera détaillée dans les sections suivantes. Une définition plus détaillée sur ces différentes situations est présentée en annexe.

D'après le tableau, trois catégories d'interaction peuvent être définies :

- *Indifférences :* les agents sont complètement indépendants, leurs buts sont compatibles, les ressources et les compétences suffisantes.
- *Coopération :* lorsque les buts des agents sont compatibles, mais que les ressources ou les compétences d'un ou plusieurs agents sont insuffisantes, les agents doivent coopérer afin d'atteindre leurs buts.
- *Antagonisme :* les buts des agents sont incompatibles. Si les ressources sont suffisantes, les agents se trouvent en situation de compétition. Sinon, la situation d'interaction devient une situation de conflit.

Ces situations d'interactions peuvent être affinées en décrivant d'autres paramètres beaucoup plus détaillés, tels que la reconnaissance des buts communs les désires, les croyances, etc. Les situations d'interaction peuvent ainsi être analysées de manière hiérarchique. En effet, une situation d'interaction complexe est composée de situations plus élémentaires. On peut ainsi distinguer des macros situations d'interaction qui sont caractéristiques d'une analyse globale de l'activité d'un ensemble d'agents et des micros situations qui se situent à un niveau de détail plus fin.

1.4.3 Interaction directe ou indirecte :

En général, les interactions sont mises en œuvre par un transfert d'informations entre agents ou entre l'environnement et les agents, soit par perception, soit par communication. Par la perception, les agents ont connaissance d'un changement de comportement d'un tiers au travers du milieu. Par la communication, un agent fait un acte délibéré de transfert d'informations vers un ou plusieurs autres agents. L'interaction peut être décomposée en trois phases non nécessairement séquentielles [Maudet et Chaïb-draa, 1996] :

- La réception d'informations ou la perception d'un changement,
- Le raisonnement sur les autres agents à partir des informations acquises,
- Une émission de message(s) ou plusieurs actions modifiant l'environnement. Cette phase

est le résultat d'un raisonnement de l'agent sur son propre savoir-faire et celui des autres agents.

Interaction indirecte :

Définition 14: Interaction indirecte selon Keil et Goldin[Keil et Goldin, 2003]

« Une interaction indirecte est observable via des changements d'état persistants ». Les agents interagissent par l'intermédiaire des actions qu'ils font sur l'environnement, et qui sont visibles pour les autres agents. Une interaction est qualifiée d'indirecte si :

- Elle est médiée par l'environnement.
- N'est pas directement dirigée vers un autre agent

Les agents participant à cette interaction sont ceux capables de percevoir ces changements de l'environnement. Ainsi un agent en effectuant une interaction indirecte, n'est pas certain de savoir avec quel(s) autre(s) agent(s) il est en train d'interagir.

Le schéma général d'une interaction indirecte est comme suit : un agent A émis une action ; cette action modifie l'environnement ; cette modification de l'environnement a pour conséquence une modification des perceptions des agents voisins, comme cela peut être le cas pour B (éventuellement à partir d'un certain temps du fait de la localité des perceptions), B émet une action à partir de ses nouvelles perceptions.

Interaction directe :

Elle est dirigée explicitement vers un destinataire dans le but de modifier son comportement interne, ne fais pas intervenir directement l'environnement. En fonction des types des agents impliqués, l'interaction directe peut aussi prendre de nombreuses formes [Anne, 2002]:

- Pour des agents cognitifs, elle peut s'exprimer à l'aide de langage et de protocoles de communication élaborés (comme KQML).
- Pour les agents réactifs, à l'aide d'échange de signaux simples (comme dans le cas de l'éco résolution).

Comme nous nous focalisons sur des agents réactifs, les interactions directes que nous seront amenés à considérer, seront fondées sur des échanges de signaux simples.

1.5 Organisation des interactions :

Définition 15: *Organisation selon Morin [Morin, 1977]*

« Une organisation peut être définie comme un agencement de relations entre composants ou individus qui produisent une unité, ou système, doté de qualités inconnues au niveau des

composants ou individus. L'organisation lie de façon interrelationnelle des éléments ou événements ou individus divers qui, dès lors deviennent les composants d'un tout. Elle assure solidarité et solidité relatives, donc assure au système une certaine possibilité de durée, en dépit de perturbations aléatoires ».

Les structures organisationnelles peuvent être constituées de deux manières :

1. Elles peuvent être définies à priori par le concepteur, et l'on parle alors d'organisations prédéfinies, ce qui signifie que les relations abstraites, qu'elles soient statiques ou dynamiques est déterminé à l'avance. Dans le cas d'organisations évolutives, c'est l'ensemble des relations abstraites possibles et l'ensemble des transformations qui sont connues à l'avance. On pourra se référer à une première analyse des organisations évolutives prédéfinies.
2. Elles peuvent aussi être définies à posteriori, et l'on parle alors d'organisations émergentes. Celles-ci, qui ne comprennent le plus souvent que des agents réactifs, sont caractérisées par l'absence de structure organisationnelle prédéfinie, celle-ci résultant totalement des interactions entre agents. Dans ce cas, les positions et les relations ne sont pas déterminées à l'avance, mais apparaissent comme le produit des comportements de chacun des agents. Plus exactement, la répartition des fonctions et des tâches suit un processus d'auto-organisation, qui permet à une organisation d'évoluer et de s'adapter facilement aux modifications de l'environnement et aux besoins d'un groupe d'agents.

Bien qu'il existe aussi une certaine forme d'émergence dans les organisations prédéfinies, variables ou évolutives, cette *émergence* est toujours contrôlée par des schémas de communication bien établis, et les organisations qui en résultent sont toujours décrites par des structures organisationnelles très précises. On peut distinguer trois niveaux d'organisation dans les SMA [Ferber, 1995]:

1. *Le niveau microsocial :* où l'on s'intéresse essentiellement aux interactions entre agents, et aux différentes formes de liaison qui existent entre deux ou un petit nombre d'agents. C'est à ce niveau que la plupart des études ont été généralement entreprises en intelligence artificielle distribuée.
2. *Le niveau des groupes :* où l'on s'intéresse aux structures intermédiaires qui interviennent dans la composition d'une organisation plus complète. A ce niveau, on étudie les différenciations des rôles et des activités des agents, l'émergence de structures organisatrices entre agents et le problème général de l'agrégation des agents lors de la

constitution d'organisations.

3. *Le niveau des sociétés globales (ou populations):* dont l'intérêt se porte surtout sur la dynamique d'un grand nombre d'agents, ainsi que sur la structure générale du système et son évolution. Les recherches se situant dans le cadre de la vie artificielle se positionnent assez souvent à ce niveau.

1.6 Coordination dans les SMA :

Il y a interaction entre les agents soit parce qu'ils coopèrent, soit parce qu'ils sont en compétition. Dans les deux cases, une coordination peut être nécessaire pour améliorer le fonctionnement global du système.

La coordination est une question centrale pour les SMA et la résolution des systèmes distribués. En effet, sans coordination, un groupe d'agents peut dégénérer rapidement en une collection chaotique d'individus. Une façon simple pour assurer un comportement cohérent du groupe d'agents, serait de faire un agent centralisateur qui pourrait créer des plans d'action, et assigner les tâches aux divers agents du groupe. La difficulté de cette approche est que l'agent centralisateur puisse tenir compte des buts, des connaissances et des activités de chaque agent, la charge de communication serait énorme. Une deuxième tentative permet de résoudre le problème directement par les agents concernés [Anne, 2002].

1.6.1 Définitions :

Définition 16 : *Coordination selon Malone [Malone, 1988]*

Une coordination est : *« L'ensemble des activités supplémentaires qu'il est nécessaire d'accomplir dans un environnement multi-agents et qu'un seul agent poursuivant les mêmes buts n'accomplirait pas »*

Selon J.Ferber [Ferber, 1995] la coordination des actions est l'une des principales méthodes pour assurer la coopération entre agents autonomes, elle est définie comme suit :

Définition 17 : *Coordination selon Ferber [Ferber, 1995]*

« L'articulation des actions individuelles accomplisse par chacun des agents de manière à ce que l'ensemble aboutisse à un tout cohérent et performant ».

La coordination d'actions est nécessaire pour quatre raisons principales [Ferber, 1995] :

1. Les agents ont *besoin d'informations et de résultat* que seuls d'autres peuvent fournir.
2. *Les ressources sont limitées :* les ressources sont limitées, si elles sont réduites et

plusieurs agents ont besoins de les utiliser, qu'il s'agisse d'espace, d'énergie et d'outils. La coordination est d'autant plus importante que les ressources sont faibles.

3. *Optimiser les couts :* tout en éliminant les actions inutiles, et en évitant les redondances d'action.

4. Permettre à des agents ayant des *objectifs distincts, mais dépendants* les uns des autres de satisfaire ces objectifs et d'accomplir leurs travaux, en tirant éventuellement parti de cette dépendance.

1.6.2 Problèmes de coordination d'actions :

Lorsqu'il y a plusieurs agents plongés dans un même environnement, et qui doivent accomplir des objectifs simultanément, ils se trouvent évidemment dans une situation de coopération, et plusieurs cas peuvent se présenter. Si les agents ont des buts indépendants, les seuls problèmes qu'ils peuvent rencontrer viennent des conflits éventuels d'accès à des ressources communes. On peut alors distinguer deux grandes catégories de situations :

1. Les premiers ont trait à l'accès d'une même portion d'espace par un ensemble d'agents. Il s'agit alors pour eux d'accomplir leurs buts tout en s'évitant. Les techniques les plus utilisées dans ce cadre sont celles qui se rapportent à l'utilisation de champs de potentiels, définissant des comportements attractifs et répulsifs.
2. Les secondes portent sur l'utilisation d'une ressource ponctuelle, telle qu'un outil commun. Il est alors possible d'utiliser des mécanismes de synchronisation.

Un problème se pose lorsque les agents ont des buts dépendants, et que les actions des uns peuvent améliorer celles des autres, et donc augmenter les performances du groupe tout entier. Le principe général consiste à utiliser les capacités des agents réactifs à réagir aux modifications de l'environnement, et souvent à marquer cet environnement pour coordonner les actions des agents entre eux.

1.6.3 Formes de Coordination :

Il existe quatre formes principales de coordination d'actions [Ferber, 1995] :

1. *Coordination par synchronisation :* on parle de synchronisation lorsqu'il s'agit de gérer la simultanéité de plusieurs actions, et de vérifier que le résultat des opérations soit cohérent. Les principales solutions à ce problème sont issues des systèmes distribués ; on peut parler alors de synchronisation par mouvement ou synchronisation d'accès à une ressource.

2. *Coordination par planification :* l'accomplissement de cette coordination passe par deux étapes : la remise de l'ensemble des actions à effectuer pour atteindre un but, on produisant ainsi un ensemble de plans, puis on choisit l'un de ces plans pour qu'on exécute. Les différents plans élaborés par les agents peuvent entrainer des conflits d'objectifs ou d'accès à des ressources. Il faut alors coordonner les plans, de manière à résoudre ces conflits et satisfaire ainsi les buts des différents agents.
3. *Coordination par réglementation :* son principe est de donner des règles de comportement visant à éliminer les conflits possibles.
4. *Coordination réactive :* fait appel à la liaison perception-action d'un agent, c'est-à-dire qu'elles s'accomplie in situ, et non à priori. Les agents réactifs sont très simples et ne possèdent pas de représentation de leur environnement ; de ce fait, toutes les informations relatives à leur comportement se trouvent dans l'environnement, et leurs réactions dépend uniquement de la perception qu'ils peuvent en avoir.

Les techniques de coordination réactives se résument à l'utilisation de quelques techniques essentielles :

- Les champs de potentiels pour la détermination du déplacement des agents mobiles.
- Utilisation des marques pour coordonner l'action de plusieurs agents ; ces marques permettent d'utiliser l'environnement comme système de communication souple, robuste et simple.

L'éco-résolution est une autre méthode de résolution des problèmes de coordination, qui prend le contre-pied des techniques classiques. Plutôt que de formaliser un problème de manière globale, et de définir ensuite des méthodes de résolution s'appliquant directement à sa définition, il est préférable de reformuler le problème de manière à ce qu'il soit conçu comme un ensemble d'agents en interaction tentant de satisfaire individuellement leurs propres buts.

1.7 Problématique de prise de décision dans les SMA :

Les notions d'observabilité et d'incertitude, pour la prise de décision sont deux notions qui occuperont une place importante dans la suite de notre travail ; les sections qui suivent présenteront en détail ces deux notions.

1.7.1 Observabilité :

Nous parlerons d'observabilité partielle quand toutes les informations nécessaires à la prise de

décision, par un agent ou un groupe d'agents ne sont pas accessibles instantanément. Les agents seront alors contraints de faire face à ce manque d'information, et devront décider au mieux comment agir en fonction des informations accessibles, et des éventuelles connaissances complémentaires dont ils disposent.

Deux types d'observabilité sont envisageables, dans le cadre des systèmes multi-agents, et selon les informations auxquels les agents ont accès. L'observabilité de l'état global du système désigne l'état du SMA (agent+environnement); on distingue l'observabilité collective, qui résulte de l'agrégation des observations de tous les agents dont l'état global du système peut être collectivement partiellement observable où il n'est pas possible de déduire l'état du système, même si on a accès à toutes les observations de tous les agents ; ou collectivement totalement observable, où on peut déduire l'état global du système à partir des observations locales, et l'observabilité individuelle, où l'agent connait l'état global du système à partir de ses observations. L'observabilité de l'état local de l'agent qui sera défini selon le degré de délibération de l'agent, peut être partiellement observable où l'agent n'a pas accès à toutes les informations nécessaires pour savoir dans quel état local il se trouve ; ou complètement observable, où l'agent peut à partir de ses observations, déduire avec certitude son état local. Enfin, lorsque les agents n'ont aucune observabilité, on parle de non-observabilité. La figure2 présente les différents types d'observabilités précédemment définis.

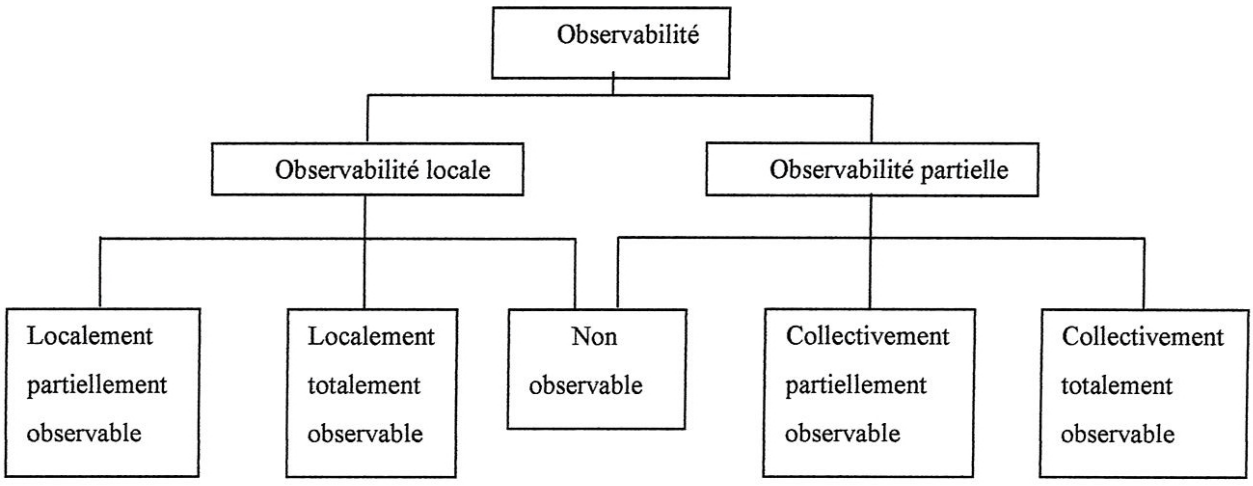

Figure 1.1 : Différents types d'observabilité [Beynier, 2006]

Si nous connaissons tout sur les états de l'environnement, les états des autres agents et de leur comportement. Il sera assez simple de décider comment agir, dés lors, le problème de décision ne se posera que dans le cas contraire.

1.7.2 Incertitude :

Il existe trois causes d'incertitude dans un système multi-agents :

- *Imprécision et limitation des capteurs :* L'imprécision des capteurs entraîne des incertitudes sur l'information perçue par l'agent, et par conséquent sur l'environnement. En effet, les capteurs peuvent être bruités ou avoir des capacités limitées, ce qui induit des erreurs dans les observations réalisées. De plus, chaque capteur est restreint par une portée qui ne lui permet généralement pas de percevoir tout l'environnement. Tout ce qui est hors de portée des capteurs, se trouve alors non observable et par conséquent incertain.
- *Imprécision des effecteurs et modélisation incomplète de l'environnement :* L'imprécision des effecteurs, ainsi que l'incomplétude de la modélisation de l'environnement, entraîne des incertitudes sur le résultat des actions des agents. L'incomplétude de la modélisation est essentiellement due au concepteur pour lequel il est, de manière générale, impossible de modéliser parfaitement l'environnement dans lequel les agents évoluent. Le concepteur peut, en effet, avoir une connaissance incomplète de tous les phénomènes et lois de l'environnement. De plus, même si le concepteur a accès à toutes ces connaissances, il est généralement impossible de rendre compte de tous les paramètres intervenant dans l'évolution du système. Les environnements dans lesquels évoluent les agents, sont alors considérés comme stochastiques, et les incertitudes sont représentées à l'aide de distributions de probabilités.
- *Les interactions entre agents :* chaque agent n'a généralement qu'une observabilité partielle des états des autres agents, et de leur comportement. Cependant, le résultat de l'exécution d'une action dépend, en partie, des actions entreprises par les autres agents. En effet, les agents évoluent dans un même environnement ; leurs actions peuvent donc entrer en interaction, conduisant alors à des effets non souhaités ou non envisagés.

1.7.3 Délibération et décision dans les SMA:

Dans le cadre de ce manuscrit, nous nous intéresserons au processus de délibération des agents, et plus particulièrement à la prise de décision. Nous considérerons par la suite des systèmes multi-agents réactifs, qui incluent la notion de coopération. L'observabilité des agents est partielle, et le résultat des actions est incertain, c'est pour cela qu'il est difficile, pour les agents de se coordonner ; on suivra l'approche où chaque agent devra décider de manière autonome, comment agir dans un environnement incertain, tout en faisant face au

manque d'observabilité, nous nous baserons pour cela, sur les processus décisionnels du Markov et ses extensions (les Interac-DEC-POMDP) qui seront définies en détail dans le chapitre suivant.

1.8 Avantages et Défis des SMA :

Les SMA sont des systèmes idéaux pour représenter des problèmes possédant de multiples méthodes de résolution, de multiples perspectives et/ou de multiples résolveurs. Ces systèmes possèdent les avantages traditionnels de la résolution distribuée, et concurrente de problèmes, comme la modularité, la vitesse (avec le parallélisme), et la fiabilité (due à la redondance). Ils héritent aussi des bénéfices envisageables de l'Intelligence Artificielle comme le traitement symbolique (au niveau des connaissances), la facilité de maintenance, la réutilisation et la portabilité, mais surtout, ils ont l'avantage de faire intervenir des schémas d'interaction sophistiqués. Les types courants d'interaction incluent la coopération (travailler ensemble à la résolution d'un but commun), la coordination (organiser la résolution d'un problème de telle sorte que les interactions nuisibles soient évitées, ou que les interactions bénéfiques soient exploitées), et la négociation (parvenir à un accord acceptable pour toutes les parties concernées).

Bien que les SMA offrent de nombreux avantages potentiels, ils doivent aussi relever beaucoup de défis. Voici les problèmes inhérents à la conception et à l'implémentation des SMA, d'après [Bond et Gasser, 1988], [Franklin et Gasser, 1997], [Iglesias et al, 1997]:

- Comment formuler, décrire, décomposer, et allouer les problèmes, et synthétiser les résultats?
- Comment permettre aux agents de communiquer et d'interagir? Quoi et quand communiquer?
- Comment s'assurer que les agents agissent de manière cohérente i) en prenant leurs décisions ou actions, ii) en gérant les effets non locaux de leurs décisions locales, et iii) en évitant les interactions nuisibles?
- Comment permettre aux agents individuels de représenter et raisonner sur les actions, plans et connaissances des autres agents afin de coordonner avec eux? Comment raisonner sur l'état de leurs processus coordonnés (comme l'initialisation ou la terminaison)?
- Comment reconnaître et réconcilier des points de vue disparates et les intentions conflictuelles dans un ensemble d'agents essayant de coordonner leurs actions?

- Comment trouver le meilleur compromis entre le traitement local au niveau d'un seul agent et le traitement distribué entre plusieurs agents (traitement distribué qui induit la communication)? Plus généralement, comment gérer la répartition des ressources limitées?
- Comment éviter ou du moins amoindrir un comportement nuisible du système global, comme les comportements chaotiques ou oscillatoires?
- Comment concevoir les plates-formes technologiques et les méthodologies de développement pour les SMA?

Ces nombreux et complexes problèmes, existent pratiquement dans tous les types de systèmes multi-agents. Les problèmes abordés dans notre manuscrit concernent plus spécifiquement la définition des comportements et des interactions d'agents réactifs, l'évaluation par les agents de leurs actions, la coopération et l'auto-organisation des agents, et la résolution collective des problèmes distribués.

1.9 Domaines d'applications des SMA :

Avant de conclure ce chapitre, nous tenons à réaliser une ébauche sur la diversité des applications des systèmes multi-agents, ainsi les applications des SMA sont largement diversifiées: la simulation de phénomènes complexes, la résolution de problème, et la conception de programmes [Wooldridge, 2002].

On utilise les systèmes multi-agents pour simuler des interactions existant entre agents autonomes. On cherche à déterminer l'évolution de ce système afin de prévoir l'organisation qui en résulte. Par exemple, en sociologie, on peut paramétrer les différents agents composant une communauté. En ajoutant des contraintes, on peut essayer de comprendre quelle sera la composante la plus efficace pour parvenir à un résultat attendu (construction d'un pont). Ce qui importe c'est le comportement d'ensemble et non pas le comportement individuel. Des applications existent en physique des particules (agent = particule élémentaire), en chimie (agent = molécule), en biologie cellulaire (agent = cellule), en éthologie (agent = animal), en sociologie et en ethnologie (agent = être humain).

L'intelligence artificielle distribuée est née pour résoudre les problèmes de complexité des gros programmes monolithiques de l'intelligence artificielle : l'exécution est alors distribuée, mais le contrôle reste centralisé. Au contraire, dans les SMA, chaque agent possède un contrôle total sur son comportement. Pour résoudre un problème complexe, il est en effet parfois plus simple de concevoir des programmes relativement petits (les agents) en

interaction plutôt qu'un seul gros programme monolithique.

Dans le même temps, le génie logiciel a évolué vers des composants de plus en plus autonomes. Par rapport à un objet, un agent peut prendre des initiatives, refuser d'obéir à une requête, se déplacer... L'autonomie permet au concepteur de se concentrer sur une partie humainement appréhendable du logiciel.

Diverses applications liées à internet font également appel à de tels systèmes : recherche d'informations, indexation des données et l'aide à la navigation. Nous tenons à mettre en avant l'importance des systèmes à base d'agents en robotique. En effet, les robots constituent naturellement des agents physiques équipés de capteurs et d'effecteurs, leur permettant de percevoir leur environnement et d'y agir. De manière générale, les systèmes multi-agents sont utilisés dans des systèmes distribués où il est nécessaire de gérer la décentralisation des données, des connaissances et du contrôle.

1.10 Conclusion :

Dans ce chapitre, nous avons détaillé la notion d'agent, en mettant l'accent sur les principales propriétés que doit avoir notre agent à savoir la rationalité, l'autonomie, la réactivité, la sociabilité et la mobilité des agents. Nous avons ensuite focalisé sur les SMA, nous avons développé les idées de multiplicité des entités, d'interaction, d'organisation et de coordination d'actions, ensuite on a présenté les différentes méthodes de résolution des problèmes de coordination entre des agents qui partageant le même environnement, puis on a présenté les problèmes de prise de décision dans des environnements incertains, les avantages et les défis du domaine et enfin les domaines d'application des SMA.

Nous nous intéressons ici à la résolution de problèmes d'agents réactifs dans un environnement incertain ; beaucoup de méthodes sont utilisées pour résoudre ces problèmes, soit dans le cadre des SMA réactifs ou cognitifs. Dans le chapitre suivant, nous présenterons les processus décisionnels de Markov et leurs extensions.

Chapitre II

Les Processus Décisionnels de Markov et leurs Extensions

Chapitre II

Les Processus Décisionnels de Markov et leurs extensions

Les modèles markoviens issus de la théorie de décision, permettent de représenter des problèmes de prise de décision dans l'incertain. D'une part, ils permettent de représenter un système, son exécution ainsi que les problèmes à résoudre, à partir d'une syntaxe constituée d'éléments non ambigus. D'autre part, ils constituent une première étape pour proposer des algorithmes génériques, manipulant ces éléments syntaxiques, pour construire une réponse au problème formalisé. Initialement, ces formalismes s'intéressent à des problèmes de prise de décision pour un seul agent, en utilisant les processus décisionnels de Markov. Des avancées récentes dans ce domaine, comme le formalisme DEC-POMDP portent sur la description de système, dont l'exécution est distribuée ; mais ce cadre ne préconise rien quant aux méthodes de résolution ; le problème de construction des comportements est donc extrêmement complexe.

Nous commencerons ce chapitre par la présentation des processus décisionnels de Markov (PDM), et leurs extensions dans le cas d'observabilité partielle mono-agent (POMDP), et terminerons la section par l'exposé des différentes approches de résolution de ces cadres. Nous présenterons ensuite les processus de décision markoviens décentralisés partiellement observables (DEC-POMDP), nous procédons ensuite à une définition de différentes méthodes de résolution de ces cadres; nous signalerons ensuite les anomalies des DEC-POMDP pour la construction de comportement autonome. Enfin, une explication détaillée du formalisme Interac-DEC-POMDP sera donné.

2.1 Les Processus décisionnels de Markov :

Les chaines de Markov[1] permettent de représenter la dynamique des systèmes dont l'évolution

[1] Une chaine de Markov est constituée d'un ensemble d'états S et d'une fonction de transition P indiquant la probabilité $P(s'|s)$ de passer d'un état s à un état s', il est possible de présenter ces données par un graphe ou bien par une matrice de transitions.

est régie par des lois stochastiques[2] vérifiant la propriété de Markov[3], dans lesquels l'observateur n'a aucun moyen d'influencer l'évolution du système [Sutton et al, 1999].

De tels formalismes s'avèrent non adéquats pour la représentation des problèmes dans lesquels un agent peut intervenir sur le système, alors que les PDM permettent de formaliser de tels systèmes ; ils constituent une extension des chaines de Markov.

2.1.1 Les composants d'un PDM :

Les processus décisionnels de Markov (PDM) modélisent des problèmes de prise de décision en environnement stochastique. Dans chaque état, l'agent doit décider quelle action exécuter ; il peut ainsi agir sur l'évolution du système.

Définition 1: ***Processus Décisionnel de Markov(PDM)***

Un Processus Décisionnel de Markov (PDM) est défini par un tuple $< S, A, T, R >$ *tel que [Beynier, 2006]:*

- S *: désigne un ensemble fini d'états.*
- A *: désigne un ensemble fini d'actions.*
- $T : S \times A \times S \rightarrow [0,1]$ *: est la matrice de transition ; elle donne la probabilité de passer de l'état s à l'état s' quand l'action a est exécutée.*
- $R : S \times A \rightarrow R$ *: est une fonction de récompense ; elle donne la récompense obtenue par l'agent lorsqu'il exécute l'action a à partir de l'état s.*

État et Action :

L'ensemble S désigne les configurations possibles du monde ; A désigne l'ensemble des actions que l'agent peut choisir d'accomplir, en vue de modifier l'état de son environnement. L'exécution d'une action à partir d'un état s est stochastique, et peut donc mener à différents états.

Matrice de transition :

La matrice de transition T représente la dynamique du système. Elle caractérise les réactions de l'environnement aux actions émises par l'agent.

La fonction de récompense :

Elle permet de représenter les objectifs de l'agent, et de guider son comportement. Elle définit les situations préférées, ainsi que les comportements non désirés de l'agent. Elle associe à

[2] Ce sont des systèmes vérifiant la propriété de Markov

[3] Affirme que la connaissance de l'état du système à l'instant t est suffisante pour prédire l'évolution du système à l'instant t+1.

chaque couple état-action (s, a), une valeur numérique traduisant le fait que l'exécution de l'action a soit souhaitée ou non à partir de l'état s. Suivant la valeur associée à ce couple, l'agent favorisera ou évitera l'exécution de l'action a à partir de s.

Il existe d'autres façons de définir une fonction de récompense. Il est possible de récompenser le fait de passer d'un état s à un état s'(R(s, s')). À chaque décision, l'agent reçoit donc un signal destiné à le récompenser ou le pénaliser. Le comportement de l'agent dépend de la façon dont il interprète ce signal. Généralement, l'agent cherche à maximiser la somme de ses récompenses. Soit r_t la récompense obtenue à l'étape t. Pour des problèmes à horizon fini H, à chaque étape t, l'agent cherche à maximiser ses H récompenses à venir, soit [Beynier, 2006] :

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_{t+H} \qquad (2.1)$$

Un facteur d'atténuation $\gamma \epsilon [0,1]$ est utilisé dans le cas des problèmes à horizon infini, afin de permettre à l'agent de calculer la récompense pondérée à long terme. On obtient alors :

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \qquad (2.2)$$

Ce facteur d'atténuation influence le comportement de l'agent ; s'il est proche de 0, les récompenses obtenues dans un futur lointain seront considérées comme insignifiantes, et l'agent favorisera ses récompenses à court terme. Si γ est proche de 1, l'agent considérera indifféremment ses récompenses à court et à long terme.

2.1.2 L'observabilité partielle dans les PDMs :

Dans la section précédente, nous avons présenté le cadre formel PDM, qui permet de représenter des problèmes de prise de décisions séquentielles, pour un système constitué d'un agent ayant une perception totale de son environnement. Pour traiter les problèmes de prise de décision dans le cadre des SMA pour lesquels les agents sont dotés de perceptions partielles, deux extensions des cadres Markoviens sont à considérer [Astron, 1965][Drake, 1962][SmallWood et Sondik, 1973]:

- La première, c'est la classe des POMDP qui traite la problématique d'observabilité partielle. Dans un POMDP, un agent ne peut plus distinguer avec certitude l'état dans lequel il se trouve. Il peut néanmoins tirer parti de son historique des couples observations-actions pour tenter de l'inférer (Section suivante).
- La seconde, c'est la classe des DEC-POMDP permettant de représenter des systèmes constitués de plusieurs agents (section 2.2.2).

Les Processus Décisionnels de Markov Partiellement Observables (ou POMDP) permettent de représenter les problèmes de prise de décision séquentielle, pour un agent capable de percevoir localement son environnement.

Définition 2 : *Processus Décisionnels de Markov Partiellement Observable (POMDP):*

Un Processus de décision markovien partiellement observable (POMDP) est défini par un tuple $< S, A, T, R, \Omega, O >$ *tel que [Phillipe, 2007]:*

- S *est un ensemble fini d'états*
- A *est un ensemble fini d'actions*
- $T : S \times A \times S \rightarrow [0,1]$*est la matrice de transition, donnant la probabilité de passer de l'état s à l'état s' quand l'action a est exécutée.*
- $R : S \times A \rightarrow R$ *est la fonction de récompense, donnant ainsi la récompense de l'agent lorsqu'il exécute l'action a à partir de l'état s.*
- Ω *un ensemble fini des observations de l'agent.*
- $O : S \times \Omega \rightarrow [0,1]$ *est la fonction d'observation.* $O(s, a, s', o)$ *correspondant à la probabilité que l'agent observe o, lorsque l'action a est exécutée à partir de l'état s et que le système arrive à l'état s'.*

Dans un POMDP, un agent ne peut plus distinguer avec certitude l'état dans lequel il se trouve. Il peut néanmoins tirer parti de son historique des couples observations-actions pour tenter de l'inférer ; l'agent a cependant besoin d'une mémoire à court terme pour se souvenir des résultats des expériences passées. Les relations entre chaines de Markov, chaines de Markov cachées, PDM et POMDP sont résumées par le tableau 2.1 :

	Observabilité totale	Observabilité partielle
Agent passif	Chaines de Markov	Chaines de Markov cachées
Agent décisionnel	PDM	POMDP

Tableau 2.1 : Relation entre les différents modèles Markoviens [Beynier, 2006]

Le passage d'un modèle à un autre dépend du degré d'observabilité de l'état du système, ainsi que de la finalité du modèle. S'il s'agit de se placer en tant qu'observateur de la dynamique du système (agent passif), les chaines de Markov sont utilisées. S'il est possible d'influencer l'évolution du système par l'exécution d'actions (agent décisionnel), le formalisme des processus décisionnels de Markov est employé.

2.1.3 Optimalité dans les MDPs :

Résoudre un MDP consiste à calculer le comportement d'un agent, étant donné un ensemble d'actions possibles, des lois du monde, une fonction de performance individuelle et un état de départ. En d'autres termes, pour un tuple $< S, A, T, R >$ donné, résoudre un MDP consiste à trouver parmi un ensemble de politiques, celle optimisant un critère de performance défini à partir de la fonction de récompense. Pour ce faire, il est nécessaire de déterminer une politique d'action de l'agent [Groupe PDMIA, 2005].

La notion de politique :

Une politique est une fonction décrivant comment l'agent doit agir dans chaque situation possible. La politique π d'un agent fait correspondre à chaque état s du système, une action que l'agent doit exécuter. Une politique π est une application :

$$\pi: \begin{cases} S \times A \rightarrow [O, 1] \\ (s, a) \rightarrow \pi(s, a) = pr[a_t = a \,/\, s_t = s] \end{cases} \quad (2.3)$$

Dans un POMDP, contrairement au cas du PDM il existe généralement des optima locaux. L'agent n'a pas accès à l'état du système, la politique est alors une fonction de l'ensemble d'observations Ω vers l'ensemble des actions A, soit :

$$\pi: \begin{cases} S \times \Omega \rightarrow [O, 1] \\ (\Omega, \mathrm{a}) \rightarrow \pi(\Omega, a) = pr[a_t = a \,/\, \Omega_t = \Omega] \end{cases} \quad (2.4)$$

Le système étant markovien, seul l'état courant est nécessaire pour décider quelle action réaliser. Pour chaque problème de décision formalisé en utilisant un MDP, il existe un ensemble Π de politiques possibles $\pi \in \Pi$. Résoudre un MDP consiste à trouver la meilleure politique, c'est-à-dire celle maximisant le critère de performance. Cette solution est appelée politique optimale et est noté π^*. Bellman [Bellman, 1957] à démontrer que tout MDP possédait au moins une politique optimale déterministe et stationnaire. De ce fait, la recherche de la politique peut être réalisée dans l'espace des politiques déterministes[4].

Évaluation d'une politique :

Afin d'établir un classement de politiques, et de pouvoir déterminer la meilleure d'entre elles, un critère de performance doit être spécifié. Il permet de définir une mesure d'utilité, associant

[4] Une politique déterministe associe à chaque état une et une seule action. Elle s'oppose aux politiques stochastiques, associant à chaque état différentes actions possibles, suivant une certaine distribution de probabilités.

à chaque politique π, une valeur V^{π}. En fonction de leurs valeurs, les politiques pourront alors être classées, et la politique optimale sera identifiée. Nous avons précédemment montré (équations 2.1 et 2.2) que le critère de performance d'un agent rationnel consistait à maximiser ses récompenses à long terme. La valeur d'une politique traduira donc l'espérance de récompense à long terme de l'agent [Beynier, 2006]

Dans les politiques à horizon fini $\mathcal{H}$, l'espérance de gain à partir d'un état s, et sur les $\mathcal{H}$ prochaines étapes est considérée. Nous obtenons alors :

$$\forall s \in S \quad V^{\pi}(s) = \mathrm{E}\left[\sum_{t=0}^{\mathcal{H}} r_t / s_0 = s\right] \tag{2.5}$$

Si la fonction de récompense est telle que R: S × A, alors $r_t = R(s_t, \pi(s_t))$. Dans le cas des problèmes à horizon infini, l'espérance de gain est pondérée par un facteur d'atténuation γ, d'où :

$$\forall s \in S \quad V^{\pi}(s) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t / s_0 = s\right] \tag{2.6}$$

Il faut noter que la valeur du facteur d'atténuation doit être choisie avec précaution, car elle influence le comportement de l'agent.

Le principe d'optimalité de Bellman :

Bellman [Bellman, 1957] a montré que la fonction de valeur d'une politique peut être calculée par récurrence, grâce à l'équation de Bellman suivante :

$$v^{\pi}{}_{(s)} = R\big(s, \pi(s)\big) + \sum_{s' \in S} \gamma p(s, \pi(s), s') v_t^{\pi}(s') \tag{2.7}$$

Il a montré qu'un système admettrait une solution unique à partir de laquelle une politique optimale peut être déduite. L'équation de Bellman est à la base de plusieurs algorithmes de résolution des PDMs, en particulier value itération et Policy itération que nous les présenterons par la suite [Bellman, 1957].

[Papadimitriou et Tsitsiklis, 1987] ont montrés que la résolution d'un MDP à horizon fini ou infini est un problème P-complet ; il est donc possible de développer des algorithmes efficaces pour leurs résolution, et la résolution d'un POMDP à horizon fini est un problème PSPACE (envisageable) ; en revanche, les POMDPs à horizon infini sont des problèmes indécidables.

2.1.4 Les algorithmes de résolution:

Déterminer la politique d'un PDM revient à calculer la valeur de la politique optimale ; celle-ci peut ensuite être déduite très facilement. Les sections suivantes présenteront différentes approches permettant de résoudre un MDP en fonction des données à la disposition de l'agent.

2.1.4.1 Résolution par planification :

L'agent dispose d'un modèle du monde, c'est-à-dire que l'agent dispose d'une estimation de la matrice de transition T et la fonction de récompense R. Les deux algorithmes très connus dans cette approche sont les suivants :

Algorithme Value itération :

L'algorithme value iteration (itération de valeur) est un algorithme de planification, cherchant à calculer par itérations successives, la fonction de valeur de la politique optimale π^* [Puterman, 1994]. Une fois cette fonction de valeur calculée, il est possible d'en déduire une politique markovienne déterministe optimale. L'algorithme est donné par :

Algorithme 2.1 : Value iteration

$entrées \leftarrow MDP, \gamma, \epsilon$
Initialiser arbitrairement $V_0(s), \forall s \in S$
  $t \leftarrow 0$
  Répéter
    $t \leftarrow t + 1$
    Pour tout $s \in S$ **faire**
      $V_t(s) \leftarrow max_a[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V_{t-1}(s')]$
    Fin pour
  Jusqu'à $max_s|V_t(s) - V_{t-1}(s)| < \epsilon$
  $\forall s, \pi(s) = argmax_a R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V_{t-1}(s')$
$retour \leftarrow \pi$

Algorithme Policy itération :

L'algorithme Policy itération (itération de la politique) consiste à construire une suite de politiques π convergeant vers la politique optimale π^* [Puterman, 1994]. L'algorithme 'Policy itération' se déroule en cycles de deux phases (algorithme 2.2):

- La première phase propose d'évaluer la politique π à l'instant t avec l'algorithme 'Policy évaluation'
- La seconde phase construit une nouvelle politique plus performante en fonction de la fonction de valeur de la politique précédente.

Algorithme 2.2 : Policy iteration

$cntrécs \leftarrow MDP, \gamma, \epsilon$
Initialiser π *et* π'
$t \leftarrow 0$
Répéter
  $\pi \leftarrow \pi'$
  Pour tout $s \in S$ **faire**
    Calculer $V^{\pi}(s)$
  Fin pour
  Pour tout $s \in S$ **faire**
    $\pi'(s) = argmax_a[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V_{\pi}(s')]$
  Fin pour
Jusqu'à $(V^{\pi'} = V^{\pi})$
$retour \leftarrow \pi$

2.1.4.2 Résolution par Apprentissage :

Si l'agent évolue dans un environnement totalement inconnu, il n'est généralement pas possible de modéliser T et R. Dans ce cas, l'agent peut directement apprendre sa politique en utilisant des algorithmes d'apprentissage par renforcement [Sutton et Barto, 1998].

1) *L'apprentissage par renforcement :* l'apprentissage par renforcement est qualifié d'apprentissage semi-supervisé, où l'agent ne reçoit pas de modèle des sorties attendues ; au cours de ces expériences un agent reçoit une note, la récompense lui donnant un aperçu de la pertinence de son action. Il peut alors apprendre par essai-erreur à corriger son comportement optimal sans aide extérieure.
2) *Q-Learning :* se fond sur la fonction état-action, où Q-valeur associe à chaque politique π l'avantage des fonctions action-valeur est qu'elles permettent de trouver la politique optimale, sans recours au modèle du monde. Ainsi, l'avantage du Q-Learning est de pouvoir apprendre une politique, tout en suivant d'autres politiques, et en s'assurant que chaque paire (état, action) peut être visiter un nombre infini de fois.
3) *Dilemme exploration/exploitation :* un agent doit explorer son environnement pour acquérir le plus d'informations possible, et décider correctement des actions à entreprendre. Le dilemme exploration/exploitation se pose pour un agent, dans la mesure où il doit choisir entre continuer à exécuter une séquence d'actions pour recevoir une récompense, ou aller explorer l'environnement pour découvrir des récompenses plus importantes. Une possibilité proposée est de suivre une politique $\varepsilon - greedy$, cette politique est stochastique définie à partir des Q-valeurs de l'agent, et dépend d'un facteur ε .

$$\pi(s) = \begin{cases} argmax_a(Q(s,a) & avec\ une\ probabilité\ de\ 1-\varepsilon \\ aléatoire & avec\ une\ probabilité\ \varepsilon \end{cases} \quad (2.8)$$

2.2 Les Processus Décisionnels de Markov et les SMA:

Nous allons traiter dans ce qui suit, la résolution des problèmes de décision dans les systèmes multi-agents, nous aborderons plus particulièrement l'utilisation des processus décisionnels de Markov pour la résolution des problèmes de décision multi-agents.

2.2.1 Notion de politique:

Résoudre un problème de décision multi-agents consiste à calculer une politique jointe $\pi = (\pi_{1,\ldots,}\pi_n)$, où π_i correspond à la politique de l'agent i. Deux types de politiques peuvent être identifiés [Xuan et Lesser, 2002]: les politiques centralisées qui spécifient pour chaque agent, l'action qu'il devra exécuter en fonction de l'état du système ; les politiques décentralisées qui spécifient pour chaque agent quelle action réaliser, en fonction des connaissances que l'agent a de l'état global du système. À partir des connaissances de l'agent (les observations), ce type de politiques est beaucoup mieux adapté dans le cas d'agents répartis dans l'espace et autonomes ; la politique jointe des agents est calculée avant que les agents commencent l'exécution de leurs tâches. La phase précédant l'exécution et durant laquelle le calcul des politiques est réalisé, est appelée phase « hors ligne » ; la phase d'exécution des tâches correspond à la phase « en ligne ». On s'intéressera au deuxième type de politiques (les politiques décentralisées).

Politique optimale :

Une politique optimale jointe π^* est telle qu'elle maximise l'espérance de gain des agents. π_i^* est la politique de l'agent i, la politique optimale jointe est : $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$. Dans le cadre d'agents coopératifs, la fonction de récompense est commune à tous les agents ; maximiser donc le gain d'un agent revient à maximiser le gain de tous les agents. Supposant que l'état initial est connu, la politique optimale est telle que :

$$V^{\pi}(s_0) = \arg\_max_{\pi} V^{\pi}(s_0) \quad (2.9)$$

Sachant que la valeur d'une politique est mesurée par :

$$V^{\pi}(s_0) = \mathrm{E}[\sum_{t=0}^{\mathcal{H}-1} R(s_t, \pi(s_t), s_{t+1}) \,|s_0] \quad (2.10)$$

Où : $\mathcal{H}$ désigne l'horizon du problème, s_t l'état du système à l'instant t, et a_t désigne l'action jointe exécutée par les agents à l'instant t , s_0 correspond à l'état initial et $R(s_t, \pi(s_t), s_{t+1})$ est la récompense obtenue par les agents, lorsque le système passe d'un état s_t à un état s_{t+1} en effectuant l'action a_t.

Dans les systèmes non coopératifs, chaque agent possède sa propre fonction de récompense, on obtient alors la mesure suivante :

$$V^{\pi}(s_0) = \mathrm{E}[\sum_{t=0}^{\mathcal{H}-1} R_i(s_t, \pi(s_t), s_{t+1}) \,|s_0] \qquad (2.11)$$

2.2.2 Processus Décisionnels de Markov décentralisés:

Les Processus Décisionnels de Markov Décentralisés Partiellement Observables (DEC-POMDP), constituent une extension des POMDPs pour des domaines où le contrôle est décentralisé, et dans lesquels les agents n'ont qu'une vision partielle de leurs environnements [Beynier, 2006]. Ces formalismes permettent la modélisation des problèmes de décision multi-agents, dans lesquels des agents doivent réaliser un ensemble de tâches, en maximisant leur récompense globale. De plus, chaque agent doit décider de manière autonome quelle action exécuter, tout en n'ayant qu'une vue partielle du système. Les DEC-MDPs et les DEC-POMDPs définissent un ensemble d'observations Ω, constituées des observations locales Ω_i des agents, une fonction d'observation O est également définie; elle décrit la probabilité qu'un agent i observe o_i étant donné un état de départ s, une action jointe a et un état d'arrivée s'.

Définition 3 : ***les Processus Décisionnels de Markov Décentralisés Partiellement Observables***

Un DEC-POMDP est un tuple $< S, Ai, T, o_i, O, R >$ *tel que [Beynier, 2006]:*

- S *désigne l'ensemble des états possibles du monde,*
- A_i *l'ensemble des actions possibles pour l'agent i. Une action jointe* $a = (a_0, a_1, \ldots, a_n)$ *est un tuple, constitué de l'ensemble des actions émises par les n agents. L'espace des actions jointes est constitué par l'ensemble de ces tuples* $A = \times A_i$
- $T : S \times A \times S \rightarrow [0,1]$ *définit la matrice de transition du système,*
- o_i *est l'ensemble des observations possibles pour l'agent i,*
- $O: S \times A \times o_i \rightarrow [0,1]$ *définit les probabilités d'observation (sur les actions et observations jointes),*
- $R : S \times A \rightarrow R$ *désigne la fonction de récompense globale du système,*

La fonction de transition, ainsi que la fonction de récompense dépendent de l'action jointe exécutée par tous les agents. Les POMDPs, les DEC-POMDPs sont définis sur un horizon H, fini ou infini. À chaque exécution d'une action jointe, la fonction R détermine la récompense reçue par le système, le but des agents étant de maximiser la somme des récompenses obtenues. Notons qu'un DEC-POMDP à un seul agent, est équivalent à un POMDP. Les relations entre les MDPs, POMDP, DEC-MDPs et DEC-POMDPs peuvent être résumées par le tableau 2.2 et la figure 2.1:

Type de contrôle	centralisé		Décentralisé	
État global collectivement totalement observable	oui	non	oui	non
Formalisme	MDP	POMDP	DEC-MDP	DEC-POMDP

Tableau 2.2 : Différence entre MDP, POMDP, DEC-MDP et DEC-POMDP [Beynier, 2006]

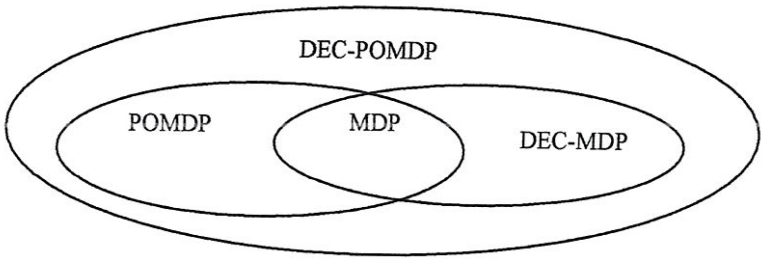

Figure 2.1 : Relation entre les cadres formels [Groupe PDMIA, 2005]

Observabilité partielle dans les DEC-POMDP:

Chaque agent ne perçoit que partiellement l'état global du système. La fonction d'observabilité O associe une probabilité $O = (o_0, o_1, \ldots, o_n)$ pour tout état global s donné ; toute action jointe $a = (a_0, a_1, \ldots, a_n)$, et toute observation jointe $O(s, a_0, a_1, \ldots, a_n, o_0, o_1, \ldots, o_n)$ donnent la probabilité que chaque agent i observe o_i en fonction de l'état global du système s et de l'action jointe a [Vincent, 2005].

Interaction dans les DEC-POMDP :

Comme la notion d'interaction est centrale dans ces systèmes, il est naturel de s'intéresser à la manière dont elle est instanciée dans ce cadre formel. Les comportements des agents sont couplés par l'intermédiaire de la matrice de transition. C'est elle qui décrit l'évolution du système en fonction de l'action jointe. Les conflits entre les agents sont résolus lors de l'évolution du système par cette même matrice. Seules des interactions indirectes médiées par

l'environnement sont possibles [Vincent, 2005]. Un agent n'a pas de vision locale des conséquences à long terme de ses actions pour deux raisons :

- Tout d'abord, il ne peut pas savoir a priori avec quel agent il va interagir, lorsqu'il modifie l'environnement, puisque ces modifications seront perçues par les autres agents en fonction de leurs comportements, et de l'état global du système.
- De plus, les conséquences réelles et ponctuelles d'une action émise par un agent, dépendent des actions émises par les autres agents. Ce qui fait que les lois du monde perçues par un agent, ne sont plus stationnaires et peuvent évoluer lorsque les autres agents modifient leurs comportements. Il est donc nécessaire pour un agent, de prendre en compte, d'une manière ou d'une autre, la présence d'autres entités dans le système afin de choisir au mieux ses actions.

L'absence de représentation explicite au niveau individuel de la notion d'interaction, pose alors problème, puisque les agents n'ont pas accès à la matrice de transition, dans laquelle les couplages entre agents sont représentés, et qu'ils ne disposent donc pas d'une structure nécessaire pour appréhender la présence d'autres agents dans le système. Ce manque de formalisme reporte ainsi de nombreux problèmes au niveau des processus de résolution.

2.2.3 Méthodes de résolution des DEC-POMDPs

Les méthodes proposées peuvent être divisées en deux catégories distinctes. La première regroupe les approches dites de résolution exacte, c'est-à-dire cherchant une solution optimale au problème. Un second type d'approches consiste à développer des algorithmes d'approximation de la solution optimale. Ces derniers calculent une solution presque optimale ou optimale localement. La complexité de tels algorithmes est généralement plus raisonnable, mais l'optimalité n'est alors plus garantie [Groupe PDMIA, 2005].

2.2.3.1 Algorithmes de résolution exacte

Parmi les approches cherchant une solution optimale, nous identifierons deux types de méthodes. Nous présenterons tout d'abord des travaux s'intéressant à la résolution de DEC-POMDPs générale.

a) Élimination itérative des politiques dominées :

Hansen et al. [Hansen et al, 2004] ont décrit un algorithme permettant de résoudre de manière optimale, des DEC-POMDPs à horizon fini H. Cet algorithme utilise la programmation dynamique, ainsi que des principes de la théorie des jeux, tels que la représentation de jeux sous forme normale et l'élimination itérative de stratégies dominées. Un DEC-POMDP est

représenté par un jeu sous forme normale, construite récursivement : la forme normale du jeu à l'étape t + 1 est déterminée à partir de sa forme normale à l'étape t. Chaque étape de construction de la solution est divisée en deux phases : la construction de la forme normale pour l'étape de décision t+1 et l'élimination des stratégies faiblement dominées. Cette dernière phase permet d'élaguer l'espace des politiques et ainsi de limiter le nombre de politiques à évaluer. Les expérimentations montrent que l'algorithme est beaucoup plus efficace que l'évaluation de toutes les politiques, en utilisant la force brute (recherche exhaustive). Cependant, le nombre de politiques à évaluer augmente fortement avec l'horizon, ce qui entraîne rapidement une limitation de la taille des problèmes traités.

b) Multi-agent A* : MAA*

Szer et al [Szer et al, 2005] ont proposé une extension multi-agent de l'algorithme de recherche heuristique A* : MAA*. Cet algorithme permet de calculer la politique optimale d'un DEC-POMDP à horizon fini ou infini, tout comme l'algorithme classique A*, MAA* procède à une construction incrémentale de la solution. Grâce au principe d'optimalité de Bellman, il est possible de calculer une solution optimale jusqu'à un horizon $t < H$. Une solution partielle au problème est ainsi obtenue. Une heuristique permet ensuite d'évaluer quelle solution paraît la plus prometteuse, pour étendre cette solution partielle : on cherche à développer la solution qui conduira au comportement le plus prometteur entre t et H. Plusieurs heuristiques ont été décrites afin de calculer rapidement une approximation des politiques entre t et H, et ainsi déterminer quelle solution doit être développée en priorité. Il est proposé de réduire le DEC-POMDP à un MDP ou à un POMDP centralisé, pouvant être résolu plus facilement. L'efficacité de l'algorithme est étroitement liée à l'heuristique utilisée. Plus l'heuristique sera précise, plus la solution sera trouvée rapidement. Dans le pire des cas, la complexité de cet algorithme est équivalente à celle de la recherche exhaustive. Les résultats expérimentaux montrent que cet algorithme est plus efficace que l'élimination itérative des politiques dominées, proposée par Hansen et al. Puisque MAA* stocke moins de politiques en mémoire. Ainsi, dans des problèmes où Hansen et al. [Hansen et al, 2004] peuvent aller jusqu'à l'horizon 4, MAA* calcule une solution jusqu'à l'horizon 5. Au-delà, MAA* se trouve également confronté à une explosion de la mémoire nécessaire à la résolution du problème.

2.2.3.2 Algorithmes de résolution approchée :

Plusieurs travaux ont porté sur la mise en place d'algorithmes permettant d'obtenir une approximation de la politique optimale.

a) Limitation de la mémoire :

Afin de pallier aux problèmes d'explosion de mémoire, dus au nombre exponentiel de politiques à stocker, Bernstein et al [Bernstein et al, 2005] ont proposé un algorithme pour la résolution de DEC-POMDPs, ne consommant qu'une quantité fixe de mémoire, et pouvant traiter des problèmes à horizon fini ou infini. La politique de chaque agent est représentée par un contrôleur stochastique à états finis. Un contrôleur dit de corrélation permet aux agents de coordonner leurs politiques. L'algorithme fonctionne par amélioration itérative des contrôleurs : à chaque itération un contrôleur est mis à jour, de façon à augmenter l'utilité jointe des agents. La convergence vers un optimum global n'est pas garantie. Cependant, il est démontré que la qualité de la solution augmente de façon monotone à chaque itération. L'algorithme atteint par conséquent un optimum local.

b) Algorithmes de coévolution itératifs :

Différents algorithmes de coévolution itératifs ont été proposés, afin de permettre l'obtention d'une politique localement optimale. Le principe de ces algorithmes consiste à améliorer la politique d'un agent, en fixant les politiques de tous les autres agents [Nair et al, 2003]. Pour un système à n agents, étant fixées les politiques de $n - 1$ agents, l'algorithme calcule la politique optimale de l'agent restant. Ce principe est répété itérativement jusqu'à ce qu'il ne soit plus possible d'améliorer aucune des politiques des agents.

c) Apprentissage par descente de gradient :

L'algorithme d'apprentissage par descente de gradient proposé par Peshkin et al [Peshkin et al, 2000] retourne également un optimum local. Cet algorithme d'apprentissage distribué, permet aux agents de déterminer individuellement quelle politique exécuter afin de maximiser la récompense globale. Chaque politique est représentée par un contrôleur à états finis. À partir d'un sous-ensemble de politiques jointes, un optimum local est déterminé. Malgré la décentralisation de l'apprentissage, les applications de cet algorithme restent limitées à des problèmes de petite taille.

d) Jeux Bayésiens

Emery-Montemerlo et al [Emery-Montemerlo et al, 2004] ont également proposé un algorithme décentralisé, pour la résolution de DEC-POMDPs. Cette approche consiste à approximer un DEC-POMDP par une série de jeux bayésiens. La politique du DEC-POMDP est obtenue par concaténation de politiques plus petites, résultant de la résolution des jeux bayésiens. L'algorithme proposé, alterne planification et exécution. À chaque étape de planification, un jeu bayésien est résolu par chaque agent afin d'obtenir la politique pour cette

Le formalisme interac-DEC-POMDP intègre cette notion d'interaction directe, et permet de représenter explicitement dans un même cadre, actions et interactions directes entre agents.

2.3.2 Description du formalisme Interac-DEC-POMDP :

2.3.2.1 Présentation du formalisme :

Définition 4 : ***Le formalisme Interac-DEC-POMDP***

Un Interac-DEC-POMDP est défini par un tuple $< S, A_i, T, R, \Gamma_i, O, I, RI, TRI >$ *[Vincent et al, 2006], sachant que :*

- *S désigne l'état global du système.*
- A_i *désigne les actions possibles pour l'agent i, A désigne l'ensemble des actions jointes.*
- *T désigne la matrice de transition du système :* $T : S \times A \times S \rightarrow [0,1]$
- *R désigne la fonction de récompense globale* $R : S \times A \rightarrow \mathbb{R}$
- Γ_i *désigne les observations possibles pour l'agent i,* Γ *désigne l'ensemble des tuples d'observation jointe des agents.*
- *O désigne la fonction d'observation* $O : S \times A \times \Gamma \rightarrow [0,1]$
- *I désigne l'ensemble des interactions possibles dans le système*
- RI_k *désigne les résultats possibles de l'interaction k*
- TRI_{kj} *désigne une matrice de transition modélisant l'évolution du système, suite à l'exécution du résultat* RI_{kj} *en fonction de l'état global du système et des agents impliqués.*

Un Interac-DEC-POMDP est constitué de deux modules : un module d'action et un module d'interaction, qui seront exécutés en série (Figure 2.2).

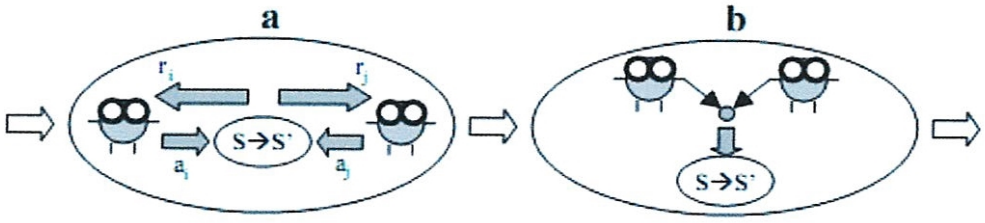

Figure 2.2 : Présentation générale de l'Interac-DEC-POMDP :
a) un module d'action et b) un module d'interaction [Vincent, 2005]

Module d'action :

L'objectif du module d'action est de pouvoir représenter :

- Les actions individuelles possibles des agents du système.

- L'utilisation des actions par les agents via leur comportement.
- L'évolution du système par rapport aux actions émises.
- La résolution des conflits entre actions.
- L'avancement de la tâche perçue localement par les agents.

La fonction de récompense se décompose à des récompenses individuelles $R = \sum_i r_i$, et que chaque agent *i* peut observer $r_i = \Gamma_i \times A_i \times \Gamma_i \rightarrow \mathbb{R}$.

Le module d'action est défini par un DEC-POMDP$< \alpha, S, A_i, T, \Gamma_i, O, R >$. Les comportements des agents sont caractérisés par leurs politiques individuelles $\pi_i: \Gamma_i \rightarrow [0, 1]$.

Algorithme d'exécution du module d'action :

L'exécution du module d'action est décrite par l'algorithme suivant [Vincent, 2005]:

Algorithme 2.3: Algorithme d'exécution du module d'action

Observation : $O = (O_0, \ldots, O_n) \leftarrow O_{(s)}$
Pour tout agent $i \epsilon [0 \ldots n]$ **faire**
| Choix de l'action i: $a_i \leftarrow \pi_i(o_i)$
Fin pour
Action jointe : $a \leftarrow (a_0, \ldots, a_n)$
Exécution de l'action jointe :$s' \leftarrow T(s, a)$
Récompense globale reçue par le système : $R \leftarrow R(s, a)$
Modification de l'état de système : $s \leftarrow s'$

Module d'interaction :

Représentation et exécution d'une interaction :

Le module d'interaction est défini par des interactions structurées de la manière suivante (figure 2.3) :

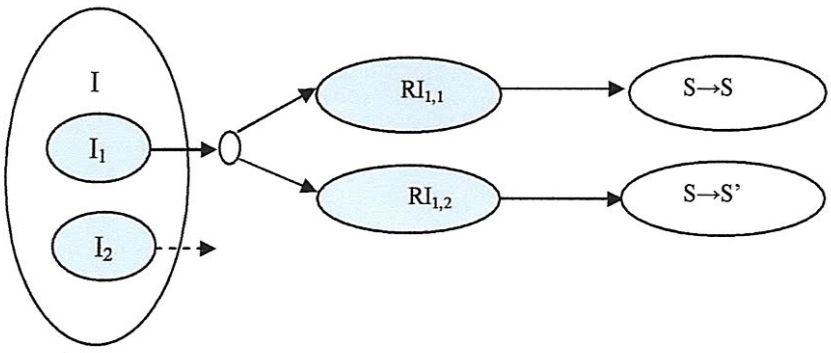

Figure2.3 : structure de l'interaction[Vincent, 2005]

- un ensemble de types d'interactions $I = \{I_k\}$
- pour chaque type d'interaction I_k, un ensemble de résultats possible $RI_k = \{RI_{k,l}\}$

- pour chaque résultat $RI_{k,l}$un ensemble de matrices de transition $TR_{k,l}$: $agent^* \times S \rightarrow S$ qui définit l'état d'arrivée du système, après application de l'interaction en fonction de l'état de départ et des agents impliqués dans l'interaction.

L'exécution d'une interaction se fait en plusieurs étapes (Figure 2.4) :

- Dans un premier temps, un agent décide de déclencher une interaction vers un ensemble d'agents particuliers, en fonction de sa politique individuelle de déclenchement $\pi_{decl,i}$: $\Gamma_i \rightarrow I \times agent^*$ (Figure2.4.a)
- Ensuite, un résultat est décidé collectivement à partir d'une politique jointe sur l'ensemble des agents impliqués Π_{agent^*}: $\Upsilon_{agent^*} \rightarrow RI_k$. Cette politique est définie pour chaque ensemble d'agents, et chaque interaction est fournit, en fonction des observations des agents impliqués, le résultat décidé. Nous verrons dans les parties suivantes comment représenter ces politiques de manière distribuée. (Figure2.4.b)
- Enfin, le résultat choisi est exécuté, et modifie l'état du système en fonction de $TR_{k,l}$(Figure2.4.c)

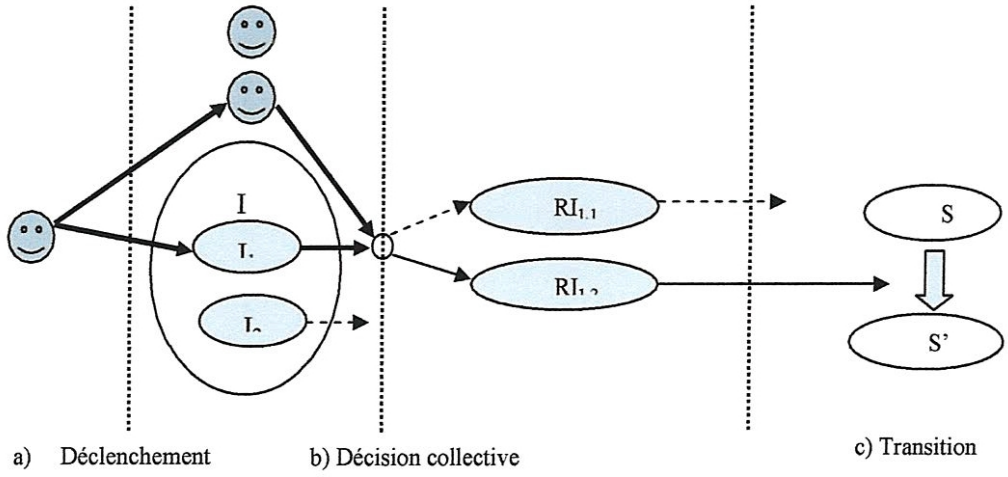

Figure 2.4 : exécution d'une interaction [Vincent, 2005]

a) Phase de déclenchement ; b) Phase de résolution ; c) Phase d'exécution du résultat (transition)

La fonction possible : c'est une fonction correspondant à la réponse du système, à un déclenchement d'une interaction. Elle associe à tout état global, toute interaction et tout tuple d'agents, un booléen qui détermine si l'interaction est envisageable (booléen retourné est vrai et l'interaction est exécutée) ou pas (booléen retourné faux et l'interaction échoue) [Vincent, 2005].

Algorithme d'exécution d'une interaction :

Algorithme 2.4: Algorithme d'exécution d'une interaction

Observation de l'agent i: $O_i \leftarrow O(s)_i$
Choix de l'interaction : $(I_k, agent_j) \leftarrow \pi_{decl,i}(O_i)$
Si *possible* $(I_k, agent_j)$ **alors**
  Observation de l'agent : j: $O_j \leftarrow O(s)_j$
  Echanges locaux et choix du résultat de l'interaction : $(RI_{k,l} \leftarrow \Pi_{i,j}(o_i, o_j))$
  Exécution du résultat : $S \leftarrow TRI_{k,l}(S, i, j)$
  Fin de l'interaction
Fin si

2.3.2.2 Exécution d'un Interac-DEC-POMDP :

L'exécution globale du modèle se fait selon des cycles, en deux phases. La première phase concerne le module d'action, la seconde concerne le module d'interaction.

Module d'action : Chaque agent i choisit individuellement son action a_i en fonction de sa politiqueπ_i. Les actions sont émises simultanément, et le système évolue conformément à la matrice de transition et à l'action jointe comme pour un DEC-POMDP.

Module d'interaction : Séquentiellement et dans un ordre aléatoire, chaque agent déclenche une interaction (potentiellement aucune) en fonction de sa politique individuelle $\pi_{decl,i}$. En déclenchant une interaction, il désigne le type d'interaction ainsi que les agents avec lesquels il souhaite interagir. L'interaction est résolue grâce aux politiques jointes de résolution d'interactionΠ ; le résultat choisi est exécuté et un autre agent est sollicité pour le déclenchement d'une interaction.

Résoudre un Interac-DEC-POMDP consiste alors à déterminer :

- Quoi faire par les politiques d'actions individuelles π_i.
- Quand, comment et avec qui interagir par les politiques de déclenchement individuelles $\pi_{decl,i}$
- Comment résoudre l'interaction par les politiques jointes de résolution Π pour toutes les interactions et tous les ensembles d'agents.

Algorithme de résolution [Vincent, 2005]:

Algorithme 2.5: Algorithme de résolution d'un Interac-DEC-POMDP pendant t_{fin} de temps

$t \leftarrow 0$
Répéter
  Module d'action
    Observation : $O = (O_0, \ldots, O_n) \leftarrow O_{(s)}$
    Pour tout agent $i \epsilon [0 \ldots n]$ **faire**
      Choix de l'action i: $a_i \leftarrow \pi_i(o_i)$
    Fin pour
    Action jointe : $a \leftarrow (a_0, \ldots, a_n)$
    Exécution de l'action jointe :$s' \leftarrow T(s, a)$
    Récompense globale reçue par le système : $R \leftarrow R(s, a)$
    Modification de l'état de système : $s \leftarrow s'$
  Module d'interaction
    Liste aléatoire des agents :$liste \leftarrow ordre_{aléatoire}$
    Pour tout $agent \in liste$ **faire**
      Observation de l'agent : i: $O_i \leftarrow O(s)_i$
      Choix de l'interaction : $(I_k, agent_j) \leftarrow \pi_{decl,i}(O_i)$
      Si *possible* $(I_k, agent_{,j})$ **alors**
        Début de l'exécution de l'interaction
        Observation de l'agent : j: $O_j \leftarrow O(s)_j$
        Echanges locaux et choix du résultat de l'interaction : $(RI_{k,l} \leftarrow \Pi_{i,j}(o_i, o_j))$
        Exécution du résultat : $S \leftarrow TRI_{k,l}(S, i, j)$
        Fin de l'exécution de l'interaction
      Fin si
    Fin pour
  $t \leftarrow t + 1$
Jusqu'à $t > t_{fin}$

2.4 Conclusion:

Dans ce chapitre nous avons présenté différents modèles stochastiques, permettant de raisonner sous incertitude ; nous avons présenté tout d'abord les processus décisionnels de Markov, ses composants, les différentes méthodes de résolution de ces cadres, ensuite on a passé à une représentation des POMDP, qui sont des extensions des PDMs utilisés dans le cadre de l'observabilité partielle des agents ; ensuite on a présenté les DEC-POMDP, qui sont aussi des extensions des POMDPs pour le cas des systèmes multi-agents avec des agents ayant des observations partielles et un contrôle autonome. Comme les DEC-POMDP ne préconisent rien quant aux méthodes de résolution, le problème de construction des comportements est donc extrêmement complexe ; pour cette raison la une extension des DEC-POMDP est réalisée ; ce nouveau formalisme est caractérisé par des prises de décision locales qui intègre explicitement au niveau individuel, une instance d'un concept d'interaction.

Dans la dernière section, on a exposé en détail le nouveau formalisme Interac-DEC-POMDP, on a ainsi présenté les différentes parties du formalisme, à savoir : le module d'action et d'interaction, la représentation et l'exécution d'une interaction. On a achevé le chapitre par l'algorithme de résolution d'un Interac-DEC-POMDP comme il a été présenté dans [Vincent, 2005].

Chapitre III

Apprentissage dans les Systèmes Multi-Agents

Chapitre III

Apprentissage dans les Systèmes Multi-Agents

La faculté d'apprentissage est nécessaire à tout système intelligent. L'apprentissage est d'ailleurs étudié dans le but d'améliorer le fonctionnement du système, en lui permettant d'évoluer. Dans les systèmes d'intelligence artificielle distribuée, le système dans son intégralité est capable d'exécuter des tâches, dont la complexité est supérieure à la somme des tâches exécutées par les agents pris individuellement. Aussi l'apprentissage est plus complexe que dans les systèmes de l'IA classiques, car il faut prendre en compte le collectif. Apprendre dans les systèmes d'intelligence artificielle distribuée et dans les systèmes multi-agents peut, être vu, soit comme un apprentissage du collectif, soit comme un apprentissage individuel à partir du groupe. L'apprentissage est une aptitude nécessaire pour amener un système à évoluer. De part le contexte d'utilisation des systèmes multi-agents, l'apprentissage semble être une faculté indispensable, tant pour l'amélioration du fonctionnement du système que pour faciliter le développement d'application. C'est pourquoi de nombreux modèles commencent à être définis. Nous présenterons ci-dessous quelques uns des plus significatifs.

3.1 Algorithmes d'apprentissage en univers Multi-Agents :

3.1.1 Le modèle de Sian :

L'apprentissage dans le système de Sian [Sian, 1991] est étudié, pour que les agents améliorent leurs connaissances sur le domaine. L'originalité de cette méthode d'apprentissage réside dans le fait qu'elle utilise la coopération pour sa mise en œuvre. Les agents confirment ou infirment, par des interactions coopératives, des hypothèses qu'ils ont déduites. Pour cela, les agents utilisent l'induction sur des exemples d'événements ayant eu lieu au sein de l'environnement, qui est identique pour tous. Le modèle présenté, appelé MALE (Multi-Agent Learning Environment), est écrit en PROLOG et est constitué :

- D'agents qui apprennent incrementalement.
- D'opérateurs qui permettent aux agents d'exprimer leurs opinions,
- D'un tableau noir pour la communication entre les agents,
- D'une fonction d'évaluation des opinions individuelles,
- D'un mécanisme d'intégration des nouvelles connaissances.

Les trois phases du cycle d'apprentissage sont :

- La génération d'hypothèses initiales par un des agents. Différents critères peuvent être définis pour le choix des hypothèses initiales à prendre en compte pour l'apprentissage (évidence, échec pour l'explication, une force associée aux croyances...).
- La coopération pour arriver à une version de ces hypothèses acceptée par tous. Celle-ci est vue en termes de représentation des connaissances, d'évaluation d'hypothèses et d'unification de toutes les opinions reçues.
- L'intégration des informations ainsi apprises par les agents. A ce niveau, le problème de la révision des croyances se pose pour intégrer les nouvelles connaissances.

Cette méthode demande donc une homogénéité au niveau de la représentation des connaissances, l'utilisation d'un langage commun pour communiquer une opinion sur une hypothèse, et la définition de fonctions d'évaluation des hypothèses. Les agents communiquent au moyen d'un tableau noir et d'un langage composé de 9 opérateurs relatifs à des hypothèses tels que :

- Changer le statut d'une hypothèse de "proposée" en "convenue",
- N'exprimer aucune opinion sur une hypothèse.

Tous les agents sont dotés d'un mécanisme d'apprentissage identique. Un agent décide du choix des hypothèses sur lesquelles va porté l'apprentissage, et sollicite les autres pour apprendre. Ainsi le mécanisme d'apprentissage est distribué, et garantit l'autonomie des agents.

3.1.2 Le modèle de Sekaran et de Sen :

Le modèle présenté dans [Sekaran et Sen, 1994] utilise l'algorithme du Q-Learning basé sur le schéma d'apprentissage par renforcement. Après s'être intéressés à l'apprentissage dans un domaine coopératif, où deux agents travaillent ensemble sur une tâche commune, sans partage explicite de connaissance ou d'information [Sen et al, 1994], Sekaran et Sen étendent leur approche à un domaine non coopératif, où les agents ont des buts conflictuels. Ces agents ne partagent ni connaissance, ni résultat. Ils apprennent une stratégie à partir d'observations, et l'améliorent avec l'expérience. L'originalité de ce travail est qu'il n'y a ni échange de connaissance explicite entre les agents, ni dépendance de relations. Cette méthode est mise en œuvre dans le monde des blocs, dans lequel deux agents autonomes ayant différentes capacités et aucune connaissance sur le domaine, doivent pousser un bloc d'une position

initiale à une position finale, dans un espace euclidien. À chaque étape, chaque agent applique une force suivant un certain angle, et le bloc est déplacé. La session est terminée dès que le bloc atteint l'une des deux positions finales, ou lorsqu'il est hors du champ de jeu. Chaque agent apprend, par l'intermédiaire de tentatives répétées, une politique optimale pour atteindre la position voulue. Le retour (feed-back) reçu après chaque action est une fonction inversement exponentielle de la distance entre la position courante du bloc et le chemin optimal associé à leurs buts respectifs. Sur la base de ces feed-back, les agents mettent à jour leur politique individuelle, en utilisant l'algorithme du Q-Learning. La convergence des matrices de stratégies (il n'y a pas de mise à jour significative des politiques individuelles sur un ensemble d'essais) est utilisée comme critère d'arrêt pour les expériences.

Le champ de force d'un agent est plus grand que celui d'un autre agent, s'il est capable de vaincre l'autre agent et de pousser le bloc vers son but. Le nombre d'essais pour converger, augmente lorsque le champ de force de l'agent le plus faible augmente (parce qu'il offre plus de résistance). Lorsque les deux agents possèdent des capacités identiques, aucun d'eux n'est capable de pousser le bloc vers son but. Ils manœuvrent alors pour pousser le bloc dans une position intermédiaire. Comme la force de l'agent le plus faible augmente, la position finale s'éloigne de la position de l'agent le plus fort. Ce phénomène est accentué lorsque le nombre d'options disponibles pour l'agent le plus faible augmente. Ceci parce que l'agent le plus fort converge vers une politique sous optimale, ce qui peut être évité en choisissant un schéma d'actions probabilistes plutôt qu'une politique déterministe.

Les auteurs montrent que les agents peuvent utiliser des schémas d'apprentissage de renforcement pour atteindre leur but, qu'ils soient coopératifs ou conflictuels, et sans avoir de modèle l'un de l'autre.

3.1.3 Le système de Mataric :

Le système multi-agent étudié [Mataric, 1994] est composé d'un groupe de 4 robots mobiles, réalisant une tâche de fourragement pour rechercher des palets dans une zone donnée, et les ramener à la "maison". Les agents n'ont pas de modèle prédéfini du monde. Ils sont tous homogènes, car construits à l'identique et n'emploient pas de modèle explicite des autres. La notion de groupe n'apporte aucun effet supplémentaire, car le comportement collectif n'est ici que la somme des activités individuelles. Les comportements sont des lois de contrôle guidées par le but, dans un domaine donné et pour atteindre un ensemble de buts. Les comportements sont fixés comme suit :

- Droite-ligne (déplacement en avant en évitant les collisions).
- Dispersion (respect d'une distance minimum entre les robots).
- Repos.
- Retour-maison.

La stratégie d'apprentissage par renforcement utilise le Q-learning, et est strictement individualiste, les agents apprennent par eux-mêmes. Les événements suivants produisent des renforcements positifs immédiats : saisir un palet, déposer un palet à la maison et dormir à la maison. Les événements suivants produisent au contraire des renforcements négatifs immédiats : déposer un palet hors de la maison, dormir hors de la maison. Lorsque le groupe de robots augmente, il y a un accroissement des interférences entre les agents, et un déclin de performance pour tous les algorithmes étudiés. Dans un scénario idéal, les autres agents devraient favoriser l'apprentissage individuel. Mais cela n'est possible que dans des sociétés où l'apprentissage est mutuel, avec des règles sociales. Ces règles sociales impliquent des comportements altruistes. Un avantage central de l'apprentissage est la possibilité d'apprendre plusieurs types de comportements en parallèle.

3.1.4 La méthode de Weib :

Weib [Weib, 1993] propose une méthode destinée à des sociétés d'agents réactifs, qui communiquent constamment avec leur environnement et entre eux par envoi de messages. Cette méthode s'inscrit dans le cadre d'un apprentissage collectif. Les agents possèdent une vision incomplète de leur environnement. Ainsi, un agent à seulement une information locale sur l'état de l'environnement, et cette information peut différer de celle que possède un autre agent. Il n'y a pas de hiérarchie entre les agents. La méthode d'apprentissage distribuée préconisée vise à une meilleure coordination entre les agents de la société. Pour cela, deux algorithmes d'apprentissage, ACE (Action Estimation) et AGE (Action Group Estimation), tous deux issus de l'algorithme du "bucket brigade" ont été développés. Le traitement réalisé par ces deux algorithmes, peut être décrit de manière simplifiée comme l'exécution répétée du cycle suivant :

- La détermination de l'action. Dans une première étape chaque agent détermine, suivant ce qu'il connaît de l'état de l'environnement, l'ensemble des actions qu'il peut effectuer.
- La compétition. Les agents concourent pour le droit de devenir actif. Cette compétition englobe le calcul et l'annonce de l'offre, ainsi que la sélection des actions qui sont actuellement effectuées.

- La répartition des crédits. Dans cette dernière étape, les agents attribuent à l'aide de la méthode du "bucket brigade", un crédit à chacun des autres agents, selon l'estimation de leurs actions.

Les deux algorithmes ACE et AGE proposés se différencient par leur manière de mettre en œuvre la seconde étape, à savoir, la compétition. Dans la méthode ACE, les agents concourent pour pouvoir effectuer des actions individuelles. Chaque agent fait une offre pour chacune des actions qu'il peut effectuer, et l'annonce à tous les autres agents qui sélectionnent alors l'offre la plus élevée parmi toutes les offres proposées. L'agent qui possède cette dernière est autorisé à exécuter l'action choisie. Dans la méthode AGE, les agents concourent pour effectuer des groupes d'actions, et non plus des actions individuelles. Pour cela, un agent estime la pertinence d'un but ou d'une action, non seulement sur la dépendance entre sa connaissance et l'état courant de l'environnement mais aussi sur la dépendance des contextes des activités possibles. Une telle séquence nécessite que les agents connaissent les autres, et qu'ils possèdent le même algorithme d'apprentissage, notamment pour le calcul des offres. Il est essentiel que les agents soient rationnels et non égoïstes, c'est-à-dire qu'il ne faut pas qu'ils insistent pour exécuter une action incompatible avec les actions précédemment exécutées. Le choix d'une offre parmi toutes celles proposées, induit également un contrôle centralisé, ce qui supprime l'autonomie de l'agent pour l'apprentissage.

Ces deux algorithmes ont été étudiés expérimentalement dans le monde des blocs. Les agents doivent, à partir d'une configuration initiale, aboutir à une configuration finale. Chaque agent est responsable d'un cube. Tout en ayant une vue limitée de l'état de l'environnement, les agents parviennent à agencer les cubes pour atteindre la configuration désirée. Cette application montre que les deux algorithmes sont capables d'étudier des séquences stables d'ensembles d'actions, et que l'algorithme AGE fournit de meilleurs résultats que l'algorithme ACE.

Discussion sur les différents modèles :

Dans ces quelques modèles, les auteurs montrent que l'activité d'apprentissage collectif permet au système d'être plus performant (du point de vue de son fonctionnement) qu'un système dans lequel l'apprentissage est réalisé de manière classique. L'étude de ces méthodes fait apparaître deux niveaux d'apprentissage : les connaissances des agents s'enrichissent soit au niveau du domaine des compétences, soit au niveau des interactions entre les agents. L'apprentissage classique en IA permet certes d'enrichir les connaissances relatives au domaine, mais le fait que l'agent interagisse avec d'autres lui permet d'apprendre plus sur le

domaine que s'il est isolé. C'est le cas dans la méthode utilisée par Sian [Sian, 1991] et par Sekaran [Sekaran et Sen, 1994]. Les performances associées au fonctionnement du système, sont aussi améliorées par le fait que les agents se coordonnent ou interagissent mieux, comme avec la méthode de Weib [Weib, 1993]. Dans la plupart des systèmes présentés précédemment, les agents sont autonomes pour la résolution, mais dans les méthodes d'apprentissage présentées par Weib, une synchronisation est requise pour apprendre. Ceci peut nuire à l'autonomie de l'agent. Ces réalisations montrent aussi que tous les agents ont tendance à utiliser le même algorithme d'apprentissage.

Dans notre travail, nous chercherons à garantir au maximum l'autonomie des agents ; pour cela on n'utilise aucun des algorithmes décrits ; on a recours à des techniques d'apprentissage par renforcement pour l'apprentissage de nos agents ; dans la section qui suivra, nous présenterons les principaux algorithmes d'apprentissage par renforcement.

3.2 Apprentissage par renforcement :

L'apprentissage par renforcement est une méthode qui permet de trouver, par un processus d'essai/erreur, l'action optimale à effectuer pour chacune des situations qu'un agent va percevoir, afin de maximiser ses récompenses. C'est une méthode d'apprentissage orientée, objective qui va conduire à un contrôleur optimal pour la tâche spécifiée par les récompenses. Cette méthode est de plus non supervisée, car la récompense ne donne pas l'action optimale à réaliser, mais simplement une évaluation de la qualité de l'action choisie. Elle permet enfin de résoudre les problèmes de récompense retardée, pour lesquels il faut apprendre à sacrifier une récompense à court terme, pour obtenir une plus forte récompense à long terme, et donc apprendre de bonnes séquences d'actions qui permettront de maximiser la récompense à long terme.

3.2.1 Définition :

Un apprentissage par renforcement désigne toute méthode adaptative permettant de résoudre un problème de décision séquentielle [Sutton et Barto, 1998]. Un paradigme classique pour présenter les problèmes d'apprentissage par renforcement, consiste à considérer un agent autonome, plongé au sein d'un environnement, et qui doit prendre des décisions en fonction de son état courant. En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative. L'agent cherche, au travers d'expériences itérées, un comportement décisionnel (appelé stratégie ou politique, et qui est une fonction associant à l'état courant,

l'action à exécuter) optimal, en ce sens qu'il maximise la somme des récompenses au cours du temps.

3.2.2 Principe de l'apprentissage par renforcement :

Formellement, la base du modèle d'apprentissage par renforcement consiste en:

1. Un ensemble d'états S de l'agent dans l'environnement.
2. Un ensemble d'actions A que l'agent peut effectuer.
3. Un ensemble de valeurs scalaires "récompenses" R que l'agent peut obtenir.

À chaque pas de temps t de l'algorithme, l'agent perçoit son état $s_t \in S$ et l'ensemble des actions possibles $A(s_t)$. Il choisit une action $a \in A(s_t)$ et reçoit de l'environnement, un nouvel état s_{t+1} et une récompense r_{t+1} (qui est nulle la plupart du temps, et vaut classiquement 1 dans certains états clefs de l'environnement). Fondé sur ces interactions, l'algorithme d'apprentissage par renforcement doit permettre à l'agent de développer une politique $\pi: S \rightarrow A$ qui lui permette de maximiser la quantité de récompenses. Cette dernière s'écrit $R = r_0 + r_1 + \ldots + r_n$ dans le cas des processus de décision markoviens (MDP), qui ont un état terminal, où $R = \Sigma t \gamma t r t$ pour les MDPs sans état terminal (où γ est un facteur de dévaluation compris entre 0 et 1 et permettant, selon sa valeur, de prendre en compte les récompenses plus ou moins loin dans le futur, pour le choix des actions de l'agent).

L'idée fondamentale de l'apprentissage par renforcement est d'améliorer une politique courante, après chaque interaction avec l'environnement. Il s'agit d'un renforcement local qui ne nécessite donc pas une évaluation complète de la stratégie. En pratique toutefois, la plupart des algorithmes d'apprentissage par renforcement ne travaillent pas directement sur la politique, mais passent par l'approximation itérative d'une fonction de valeur, issue de la théorie des PDM présentée au chapitre précédent.

La formalisation mathématique des algorithmes d'apprentissages par renforcement, telle que nous la connaissons aujourd'hui, s'est développée à partir de 1988 lorsque Sutton [Sutton, 1988] puis watkins [Watkins, 1989] ont fait le lien entre leurs travaux et le cadre théorique de la commande optimale proposée par Bellman en 1957 avec la notion de PDM [Bertsekas, 1995].

3.2.3 Méthodes d'optimisation de comportement :

Dans le cas où les agents ne disposent pas d'un modèle de la dynamique de l'environnement, ils doivent effectuer leurs apprentissage de la politique optimale en interagissant avec

l'environnement ; des conséquences de ces interactions (retours perçus et transitions effectuées), ils déduiront une estimation de la fonction de valeurs (ou de la fonction qualité) qu'ils raffineront petit à petit, d'où ils déduiront une politique. Nous nous penchons à présent sur les algorithmes qui permettent à un agent de découvrir une politique optimale. Il existe trois classes d'algorithmes d'optimisation du comportement :

1. Programmation dynamique (PD):

Les algorithmes de la programmation dynamique [Bellman, 1957][Bertsekas et Tsitsiklis, 1996] s'appliquent dans le cas où l'agent dispose d'un modèle de son environnement, c'est-à-dire lorsque les fonctions de transition P, et de récompense R sont connus à priori. Ces méthodes permettent la résolution de problèmes de planification, plutôt que le problème de l'apprentissage ; ce sont des méthodes incrémentales. Les deux algorithmes les plus connus de cette catégories sont : l'algorithme de policy iteration, et celui de value iteration présentés dans le chapitre précédent.

2. Monté Carlo (MC) :

Les méthodes de Monté Carlo [Michie et Chambers, 1986] sont des méthodes qui s'appuient sur l'expérience, cas dans lequel cas l'agent n'est pas obligé de connaitre parfaitement le modèle de l'environnement, et donc un problème d'apprentissage est posé, où T et R doivent être apprises par l'expérience ; ces méthodes ne sont pas incrémentales.

3. Méthodes de Différence Temporelle (TD) :

Les méthodes de Différence Temporelle [Samuel, 1959][Sutton, 1988] sont des combinaison ; des idées des méthodes de la Programmation Dynamique et des méthodes monté Carlo ; ce sont des méthodes qui ne nécessitent pas de modèle de la dynamique de l'environnement. Dans la section ci-dessous, nous présenterons les différents algorithmes issus de cette catégorie.

3.2.4 Principaux algorithmes d'apprentissage par renforcement :

1. TD(0) :

TD est l'algorithme de base de l'apprentissage par renforcement. Il repose sur une comparaison entre la récompense que l'on reçoit effectivement et la récompense que l'on s'attend à recevoir, en fonction des estimations construites précédemment. Les équations de mise à jour incrémentales sont :

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \qquad (3.1)$$

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t \quad (3.2)$$

Sachant que :

$V(s)$: Fonction d'évaluation des états

α : Taux d'apprentissage

δ_t : Erreur de différence temporelle

r_{t+1} : Récompense immédiate

γ : Facteur de décompte temporel

Dans le cadre de cet algorithme, l'agent ne peut pas déterminer à l'avance quel est l'état suivant, de meilleure valeur estimée ; il est donc incapable de déduire quelle politique suivre ; c'est pour quoi les algorithmes qui prennent en compte la valeur des couples état/action (s, a) sont préférés.

2. Sarsa:

Watkins [Watkins, 1989] a introduit la fonction de valeur Q, dont la donnée est équivalente à celle de V lorsqu'on connait la fonction de transition P. L'algorithme Sarsa est similaire à l'algorithme TD(0), à ceci prés qu'il travaille sur les valeurs des couples (s, a) plutôt que sur la valeur des états ; on estime la valeur d'une action à partir d'un état plutôt que d'être dans un état. Son équation de mise à jour est la suivante:

$$Q(s_n, a_n) \leftarrow Q(s_n, a_n) + \alpha\big(r_{n+1} + \gamma Q(s_{n+1}, a_{n+1}) - Q(s_n, a_n)\big) \quad (3.3)$$

L'information nécessaire pour réaliser une telle mise à jour est donc le quintuple $(s_n, a_n, r_{n+1}, s_{n+1}, a_{n+1})$, d'où découle le nom de l'algorithme. Effectuer ces mises à jour implique que l'agent détermine avec un pas de regard en avant, quelle est l'action a_{i+1} qu'il réalise lors du pas de temps suivant, lorsque l'action a_i dans l'état si l'aura conduit dans l'état s_{i+1} [Beynier, 2006].

3. Q-learning :

L'algorithme Q-Learning se présente comme une simplification de l'algorithme Sarsa par le fait qu'il n'est plus nécessaire pour l'appliquer de déterminer un pas de temps à l'avance quelle sera l'action réalisée au pas de temps suivant. Son équation de mise à jour est la suivante :

$$Q(s_i, a_i) = Q(s_i, a_i) + \alpha\big(r_{i+1} + \gamma max_{a\in A} Q(s_{i+1}, a) - Q(s_i, a_i)\big) \quad (3.4)$$

La différence entre Sarsa et Q-Learning se situe au niveau de la définition du terme d'erreur. Le terme $Q(s_{n+1}, a_{n+1})$ apparaissant dans l'équation (3.3) a été remplacé par le terme $\max_{a \in A} Q(s_{i+1}, a)$ dans l'équation (3.4) ; il apparait que l'algorithme Sarsa effectue les mises à jour en fonction des actions choisies effectivement alors que l'algorithme Q-Learning effectue les mises à jour en fonction des actions optimales. Le choix de l'action doit garantir un équilibre entre l'exploration et l'exploitation de l'apprentissage déjà réalisé. Naturellement, on conçoit intuitivement que l'exploration doit initialement être importante (quand l'apprentissage est encore très partiel, on explore), puis diminuer au profit de l'exploitation quand l'apprentissage a été effectué pendant une période suffisamment longue. Plusieurs stratégies sont utilisées classiquement :

- *gloutonne :* consiste à toujours choisir l'action estimée comme la meilleure, soit

$$a_{gloutonne} = argmax_{a \in A(s_t)} Q(s_t, a) \quad (3.5)$$

- $\epsilon - gloutonne$: consiste à choisir l'action gloutonne avec une probabilité ϵ et à choisir une action au hasard avec la probabilité $1 - \epsilon$, soit :

$$a_{\epsilon-gloutonne} \begin{cases} argmax_{a \in A(s_t) Q(s_t, a)} & avec\ probabilité\ \epsilon \\ action\ prise\ au\ hasard\ dans\ A(s_t) & avec\ probabilité\ 1 - \epsilon \end{cases} \quad (3.6)$$

- *Softmax :* consiste à choisir une action en fonction de sa qualité, relativement à la qualité des autres actions dans le même état ; il ya encore la méthode *boltzman* qui est un cas particulier de sélection softmax.

4. <u>TD(λ), Sarsa(λ), Q-learning(λ) :</u>

Dans ces méthodes, l'effet d'une récompense « r » n'est pas limité à l'état ou à la paire état-action qui vient de passer, mais est transmise aux états et actions précédents (avec un taux de diffusion$\lambda \in]0, 1[$).Cela permet d'accélérer l'apprentissage par rapport aux méthodes précédentes de différences temporelles simples, qui ne permettent cette diffusion qu'à travers un nombre bien plus grand de passages dans chaque état [Uribe et al, 1999]. Cet apprentissage est de type « on-policy » [Sutton et Barto. 1998]. Cependant, ces algorithmes présentent le défaut de ne mettre à jour qu'une valeur à pas de temps, à savoir l'état que l'agent est en train de visiter.

3.2.5 Paramètre d'apprentissage:

La formule de mise à jour de la fonction de valeur V et Q dépend de plusieurs facteurs jouant un rôle important dans cette relation : les paramètres d'apprentissage (α, γ et ε), la

récompense (r) et la valeur initiale de la fonction de valeur (ou utilité) V ou Q. pour cela il es utile détailler ces paramètres pour en déduire l'intérêt de chacun :

Le paramètre $\gamma(0 \leq \gamma < 1)$:

Appelé « facteur de décompte ou d'actualisation» ; c'est un facteur de pondération qui permet de déterminer la valeur présente d'une récompense future : une récompense reçue dans k pas dc temps vaut γ^k fois qu'elle vaudrait, si elle était reçue immédiatement. Il peut être compris donc comme un paramètre permettant de déterminer l'importance du futur dans la prise de décision. Ce critère présente un intérêt fonctionnel : il incite à maximiser les récompenses tout en diminuant le temps pour les obtenir. L'influence du temps est d'autant moins négligeable que γ est proche de 1. Au contraire, si γ vaut 0, on dit que l'agent est glouton : son but revient alors, à chaque instant, à maximiser sa récompense immédiate sans se préoccuper de celles qui suivra.

Ce facteur permet aussi d'assurer la convergence de la suite de récompenses reçues [Hubbard, 2002]; [Potapov et al, 2003]. Mais le théorème de convergence pour les algorithmes d'apprentissage, donne seulement les limites pour ces paramètres(α et γ), il ne recommande pas leurs valeurs optimales.

Le facteur α $(0 < \alpha \leq 1)$:

C'est un coefficient dénommé « taux d'apprentissage » ; il détermine l'importance de la correction réalisée sur la fonction valeur lors d'une mise à jour. Néanmoins, on peut trouver dans la littérature un certain nombre de recommandations pour le paramètre d'apprentissage « α », qui sont parfois contradictoires [Potapov et al, 2003] :

1. On le prend le plus souvent égale à 1/1+n où «n» est le nombre de visites de «s» depuis le début [Bertsekas et Littman, 1996].
2. On prend aussi α = 1/t, donc variable au cours du temps, initialement grand et tendant vers zéro [Bertsekas et Tsitsiklis, 1996].
3. Pour beaucoup de problèmes, le choix optimal de α est 1 ; selon [Kaelbling et Tsitsiklis, 1996] ceci permet d'accélérer l'apprentissage.
4. Dans [Dayan, 1992]; [Dayan et Sjnowski, 1994], les auteurs préconisent de prendre α assez petit ; ce choix est appuyé par les preuves des théorèmes de convergence.

Le renforcement r (pénalité ou récompense) :

La pénalité est celle obtenue par l'agent lorsqu'il effectue une action interdite, et la récompense est celle obtenue lorsqu'il atteint l'objectif. Ces valeurs ont une grande influence sur les résultats. Matario [Mataric, 1994] propose une méthode pour choisir des fonctions de récompenses, utilisant les connaissances implicites sur l'environnement. Cette méthode implique l'utilisation d'un processus qui permet d'enfoncer la sémantique de l'environnement dans la fonction de récompense, le plus souvent hétérogène. Il s'agit, en effet de la partie la plus sensible et la plus délicate du paramétrage de la fonction de valeur.

La fonction de valeur ou d'utilité :

Le plus souvent, on prend, zéro pour valeur initiale de V ou Q. Cependant certains auteurs, recommandent d'autres initialisations. Dans cette optique, on peut citer Mataric [Mataric, 1994], qui utilise les méthodes par estimateurs de progrès, consistant à incorporer un conseil directement dans la fonction de renforcement, ou la fonction de valeur Q pour guider l'agent. On trouve aussi les techniques par imitation [Behnke et al, 2005] ; ces dernières proposent de donner à l'agent, l'accès aux valeurs de la table Q d'un autre agent plus expérimenté.

Le paramètre d'exploration « ε » :

Au sein de l'apprentissage, l'agent explore le domaine ou l'environnement dans lequel il évolue. Cette exploration peut être entreprise de combien de manières différentes. Le niveau optimal d'exploration « ε» dépend fortement du sens du problème. Si le but est d'apprendre seulement, la valeur de « ε» peut être haute, mais si le système doit accomplir une certaine tâche tout au long de l'apprentissage, nous ne devons pas risquer d'avoir trop d'action aléatoires, mais avoir un compromis entre apprentissage et exécution de la tâche.

3.3 Conclusion

Dans le présent chapitre deux sections sont présentées, la première contient une exposition des différents algorithmes d'apprentissage dans le domaine des SMA ; dans la deuxième, la technique d'apprentissage par renforcement est présentée, avec introduction de la définition d'un apprentissage par renforcement, les différents algorithmes et des paramètres intéressants d'apprentissage par renforcement.

Dans notre travail, on se base sur un apprentissage basé sur l'algorithme de Q-Learning, donc on utilise un apprentissage par renforcement pour chaque agent ; pour passer à l'échelle collective, on utilise les heuristiques proposés dans [Vincent, 2006] à savoir l'échange des

récompenses sociales et l'échange des Q-valeurs, pour arriver à résoudre ensemble la tâche ou bien le problème initial.

Dans le chapitre suivant, on exposera l'algorithme d'apprentissage utilisé dans notre application pour l'adaptation des comportements de nos agents.

Partie A

De l'agent aux systèmes multi-agents

Conclusion

Dans cette partie, nous avons décrit les systèmes que l'on souhaite construire, elle est décomposée en deux chapitres. Tout d'abord, dans le premier chapitre on s'est intéressé au concept d'agent, en mettant l'accent sur la notion de rationalité, d'autonomie et de réactivité d'un agent ; ensuite on s'est focalisé sur les SMA et sur la notion d'interaction qui assure le lien entre les comportements individuels des agents, et le comportement global qui en résultent, en passant en revue sur la coordination des actions et les méthodes proposées pour résoudre les problèmes qui en découlent. Nous avons développé par la suite les notions d'incertitude et d'observabilité partielle, qui constituent deux propriétés principales pour nos agents. On terminera ce chapitre par les problèmes auxquels il faut faire face dans la conception des SMA, ainsi que les domaines d'application de ces systèmes.

Pour concevoir automatiquement des systèmes répondant à des problèmes collectifs, nous avons choisi de nous concentrer sur des approches formelles, ce sera le contenu de notre deuxième chapitre, que nous articulerons comme suit :

- Présentation des PDMs qui formalisent un problème de prise de décision mono-agent.
- Représentation des POMDP qui sont des PDM avec des observabilités partielles, dans un cadre mono-agent.
- Représentation des DEC-POMDP, permettant de représenter des systèmes constitués de plusieurs agents en interaction.
- Présentation des méthodes permettant de résoudre ces formalismes.
- Détail du formalisme sur lequel se base notre travail qui est le formalisme Interac-DEC-POMDP.
- Exposition de l'algorithme d'exécution du formalisme Interac-DEC-POMDP.

Enfin, on a terminé la première partie par le troisième chapitre; où on a exposé les différentes méthodes d'apprentissage développé dans le cadre des systèmes multi-agents, et on a introduit aussi les techniques d'apprentissage par renforcement. Ce qui reste à faire c'est comment concevoir les comportements des agents et quelle architecture proposer à ces agents pour leur permettre de résoudre collectivement un problème inconnu initialement ?

Partie B

Mise en Oeuvre

Partic B

Mise en oeuvre

Introduction

La présente partie contient deux chapitres essentiels, dans le premier chapitre on s'intéressera à l'exposition du processus d'apprentissage utilisé pour apprendre les comportements des agents [Vincent, 2005], on passe en revue à la représentation de l'architecture interne des agents, en expliquant ainsi les différents modules la composant, puis on explique l'application de robotique mobile qui sert de cadre expérimental à notre travail, en mettant l'accent sur un certain nombre de contraintes que doit respecter l'ensemble des robots. Enfin, on donnera le comportement global de l'agent qui lui permet d'accomplir la tache de fourragement, en le décomposant en un certains nombre de comportements élémentaires et simples à effectuer, dont la sélection d'un comportement se fait en revenant sur les Q-valeurs apprises par apprentissage.

La deuxième partie est le chapitre des résultats et simulations, on effectuera plusieurs simulations tout en faisant varier les paramètres du problème.

Chapitre IV

Conception

Chapitre IV : Conception

Dans la partie précédente, on a vu la modélisation de la conception d'un agent, en utilisant le Processus Décisionnels de Markov (Interac-DEC-POMDP), ainsi que la construction des comportements, en utilisant un algorithme d'apprentissage se basant sur l'algorithme de Q-Learning. Dans ce chapitre, nous allons exposer la structure de notre modèle de conception de l'agent, tout en définissant le principe de fonctionnement de chaque module intégré ; nous aborderons aussi l'algorithme d'apprentissage utilisé pour la construction des comportements des agents [Vincent, 2005], et nous terminerons le chapitre par l'explication de l'application sur laquelle nous allons tester cette approche.

4.1 Choix de l'algorithme:

Selon l'étude de différentes méthodes d'apprentissages exposés dans le chapitre précédent, et comme la convergence des ces algorithmes pour la résolution d'un Interac-DEC-POMDP reste non triviale, notre choix s'est orienté vers l'algorithme de résolution (aussi d'apprentissage) présenté dans [Vincent, 2005] ; Il a été introduit par Vincent Thomas en 2005 et il est sans doute un algorithme efficace. Cette caractéristique vient de sa simplicité, et ses preuves de convergence, qui ont accompagné la publication de cet algorithme [Vincent, 2005], [Vincent et al, 2006], [Vincent et al, 2004]. Dans ce qui suit nous détaillerons les constitutions de cet algorithme.

4.1.1 Représentation des politiques:

Dans un Interac-DEC-POMDP, un agent est caractérisé par trois politiques : une politique d'action, une politique de déclenchement et une politique de résolution d'interaction, ces politiques sont paramétrés par des Q-Valeurs. Dans les paragraphes qui suivent, nous expliquerons en quoi correspondent ces différentes valeurs telles quelles sont apparues dans [Vincent, 2005].

1. *Q-valeurs d'action :* ils associent à chaque agent, état individuel s et chaque action a, une valeur numérique, qui représente l'intérêt d'un agent à exécuter une action a dans un état correspondant s ; à partir de ces Q-valeurs d'actions, on construit une politique d'action consistant à choisir l'action la plus profitable d'un agent, selon la formule suivante :

$$\pi_i(s_i) = argmax_a\{Q_{action,i}(s_i, a)\} \quad (4.1)$$

2. *Q-valeurs de déclenchement :* sont associées à la politique de déclenchement ; elles associent à tout état d'agent et tout couple (interaction, agent*), un indice correspondant à l'intérêt d'un agent à exécuter cette interaction ; cet intérêt est traduit par la Q-valeur d'action maximal d'un agent après exécution de cette interaction. La politique de déclenchement d'un agent lui permet de choisir l'interaction la plus utile au système ; elle est donnée par la formule suivante:

$$\pi_{decl,i}(s_i) = \text{argmax}_{I,agent^*}\{Q_{decl,i}(s_i, I, agent^*)\} \quad (4.2)$$

3. *Q-valeurs de résolution :* chaque agent dispose de Q-valeurs d'interaction individuelles données par :

$$QInterac: I \times agent^* \times S_{agents} \times Resultats \rightarrow \mathbb{R} \quad (4.3)$$

Ces Q-valeurs d'interaction fournissent l'intérêt qu'a l'agent i à imposer le résultat$R_{k,l}$. Pour construire des politiques de résolution collectives, on suppose que les agents échangent leurs Q-valeurs d'interaction et leurs perceptions en cours d'interaction. Donc, la politique de résolution d'interaction permet la maximisation de la somme des Q-valeurs d'interaction des agents impliqués dans l'interaction, elle est donnée par :

$$\Pi_{agent^*,I_k}(S_{agents}) = \text{argmax}_{r_{k,l}}\left\{\sum_{agent^*}\left(QInterac_{agent}(I_k, agent^*, S_{agents}, R_{k,l})\right)\right\} \quad (4.4)$$

4.1.2 Apprentissage des politiques:

1. *Apprentissage des Q-valeurs d'interaction :* consiste pour un agent à évaluer l'intérêt qu'il a pour un résultat d'interaction donné ; il est traduit comme la Q-valeur d'action maximale, associée à l'état de l'agent après exécution du résultat de l'interaction. Pour répartir la tache entre plusieurs agents, il ya un échange du gain reçu par l'agent qui a le plus d'intérêt à déclencher l'interaction de façon équitable, permettant ainsi d'encourager les agents à participer à des interactions bénéfiques. Elle sera calculée en fonction du gain global[1].
2. *Apprentissage des Q-valeurs de déclenchement :* calculé à partir du gain global ; plus le gain est important plus l'agent aura tendance à exécuter cette interaction.

[1] Calculé localement par chaque agent impliqué dans l'interaction, et traduit par la différence de la somme des Q-valeurs d'actions entre avant et après l'exécution de l'interaction.

3. *Apprentissage des Q-valeurs d'actions :* sera modifié en fonction : soit de récompenses individuelles (locales) associées à l'avancement de la tache, soit parce que l'agent à contribué à un déclenchement d'interaction comme résultat de son action, il va donc recevoir une récompense sociale. L'apprentissage de ces Q-valeurs se fait par Q-Learning classique, tout en incluant les récompenses individuelles et les récompenses sociales. [Vincent, 2005]

4.1.3 Cycle d'Apprentissage:

Dans cette partie nous allons expliquer avec détail l'algorithme d'apprentissage, et présenterons la manière dont il s'insère dans l'algorithme d'exécution d'un Interac-DEC-POMDP [Vincent, 2005].

Exécution des actions :

À partir de sa politique d'action paramétrée par les Q-valeurs d'action, un agent décide d'exécuter une action dans un état s. On utilise des politiques $\varepsilon -$ greedy pour permettre à un agent d'explorer l'environnement :

$$a_i = \pi_i(s_i) = \begin{cases} aléatoire & probabilité \quad (\varepsilon) \\ argmax_a\left(Q_{action}(s_i, a)\right) & probabilité \quad (1-\varepsilon) \end{cases} \tag{4.5}$$

Le système évolue en fonction de l'action jointe de toutes les actions des agents et un agent reçoit une récompense individuelle.

Déclenchement d'interaction :

Sont des politiques $\varepsilon -$ greedy ; trois interactions sont possibles : soit un échange de bennes, soit une interaction d'accès à une case, soit il ya pas d'interaction (interaction vide).

Choix d'un résultat d'interaction :

Permet de maximiser la somme des Q-valeurs individuelles d'interaction à partir de communications locales selon la formule :

$$\begin{aligned} R_{k,l} \rightarrow \text{argmax}_{R_{k,l}}\{&\left(\text{QInterac}_1\left(I_k, \text{agent}_2, S_{\text{agents}}, R_{k,l}\right)\right) \\ &+ \left(\text{QInterac}_2\left(I_k, \text{agent}_1, S_{\text{agents}}, R_{k,l}\right)\right)\} \end{aligned} \tag{4.6}$$

Exécution du résultat :

L'interaction est exécutée et les états des agents concernés sont modifiés.

$$(s_1'', s_2'') \leftarrow TRI_{Rk}(s_1', s_2') \quad (4.7)$$

Transfert des récompenses sociales :

Le gain collectif obtenu est transféré entre les agents impliqués, qui s'expriment par la différence entre la somme des fonctions de valeur d'action des agents impliqués, avant et après l'interaction.

$$v(s) = max_a Q(s, a) \quad (4.8)$$

$$gain1 = v_{action,1}(s_1'') - v_{action,1}(s_1') \quad (4.9)$$

$$gain2 = v_{action,2}(s_2'') - v_{action,2}(s_2') \quad (4.10)$$

$$r_{s,1} = -gain1 + \frac{1}{2(gain1 + gain2)} \quad (4.11)$$

$$r_{s,2} = -gain2 + \frac{1}{2(gain1 + gain2)} \quad (4.12)$$

Apprentissage des interactions (résolution) :

Chaque agent met à jour ses Q-valeurs d'interaction en fonction de la valeur d'action de l'état après interaction comme suit :

$$\mathrm{QInterac}_1(\mathrm{I_k}, \mathrm{agent_2}, \mathrm{s'_1}, \mathrm{s'_2}, \mathrm{R_{k,l}}) = \mathrm{V_{action,1}(s''_1)} \quad (4.13)$$

$$\mathrm{QInterac}_2(\mathrm{I_k}, \mathrm{agent_1}, \mathrm{s'_1}, \mathrm{s'_2}, \mathrm{R_{k,l}}) = \mathrm{V_{action,2}(s''_2)} \quad (4.14)$$

Apprentissage des interactions (déclenchement) :

L'agent déclencheur de l'interaction modifie ses Q-valeurs de déclenchement à partir du gain global comme suit :

$$\mathrm{Q_{decl,1}(s'_1, IK, agent_2)} = (1 - \alpha)\mathrm{Q_{decl,1}(s'_1, IK, agent_2)} + (\alpha)(\mathrm{gain_1 + gain_2}) \quad (4.15)$$

Avec : α est un coefficient d'apprentissage permettant de conserver une mémoire du passé.

Apprentissage des actions :

Chaque agent met à jour sa Q-valeur d'action de départ en fonction des récompenses individuelles et des récompenses sociales ; la formule de mise à jour est indiquée ci-dessous, où $\gamma \in [0, 1]$ est le facteur de décompte :

$$Q_{action,i}(s_i, a_i) = (1-\alpha)Q_{action,i}(s_i, a_i) + \alpha\left(r_i + r_{s,i} + \gamma v(s'''_i)\right) \quad (4.16)$$

Dans la section précédente, on a expliqué les différentes parties de l'apprentissage des politiques d'action, de déclenchement et de résolution d'interactions ; dans la section suivante (algorithme 3.2) nous donnerons l'algorithme général d'apprentissage incluant ainsi les différentes parties d'apprentissages décrites précédemment ; cet algorithme est le même que celui de la résolution du formalisme Interac-DEC-POMDP avec les ajouts d'apprentissages des politiques ; ces ajouts sont marqués par des puces carrées noires.

Algorithme 4.1 : Algorithme d'apprentissage pendant t_{fin} de temps

$t \leftarrow 0$
Répéter
  Module d'action
  Observation : $O = (O_0, \dots, O_n) \leftarrow O_{(s)}$
  Pour tout agent $i \epsilon [0 \dots n]$ **faire**
    Choix de l'action i: $a_i \leftarrow \pi_i(o_i)$
  Fin pour
  Action jointe : $a \leftarrow (a_0, \dots, a_n)$
  Exécution de l'action jointe :$s' \leftarrow T(s, a)$
  Récompense globale reçue par le système : $R \leftarrow R(s, a)$
  Modification de l'état de système : $s \leftarrow s'$

  Module d'interaction
  Liste aléatoire des agents :$liste \leftarrow ordre_{aléatoire}$
  Pour tout $agent \in liste$ **faire**
    Observation de l'agent : i: $O_i \leftarrow O(s)_i$
    Choix de l'interaction : $(I_k, agent_j) \leftarrow \pi_{decl,i}(O_i)$
    Si *possible* $(I_k, agent_{,j})$ **alors**
      Début de l'exécution de l'interaction
      Observation de l'agent : j: $O_j \leftarrow O(s)_j$
      Echanges locaux et choix du résultat de l'interaction : $(RI_{k,l} \leftarrow \Pi_{i,j}(o_i, o_j))$
      - Calcul des récompenses sociales : $r_{soc,i,j}$

      Exécution du résultat : s$\leftarrow TRI_{k,l}(S, i, j)$
      - Mise à jour des Q-valeurs d'interaction
      - Mise à jour des Q-valeurs de déclenchement

      Fin de l'exécution de l'interaction
    Fin si
  Fin pour
  - **Mise à jour des Q-valeurs d'action**

  $t \leftarrow t + 1$
Jusqu'à $t > t_{fin}$

4.2 Modèle de Conception:

Le modèle de conception décrit les différents composants d'un agent, notre modèle de conception de l'agent à base du formalisme Interac-DEC-POMDP, est représenté par le schéma suivant (figure 4.1):

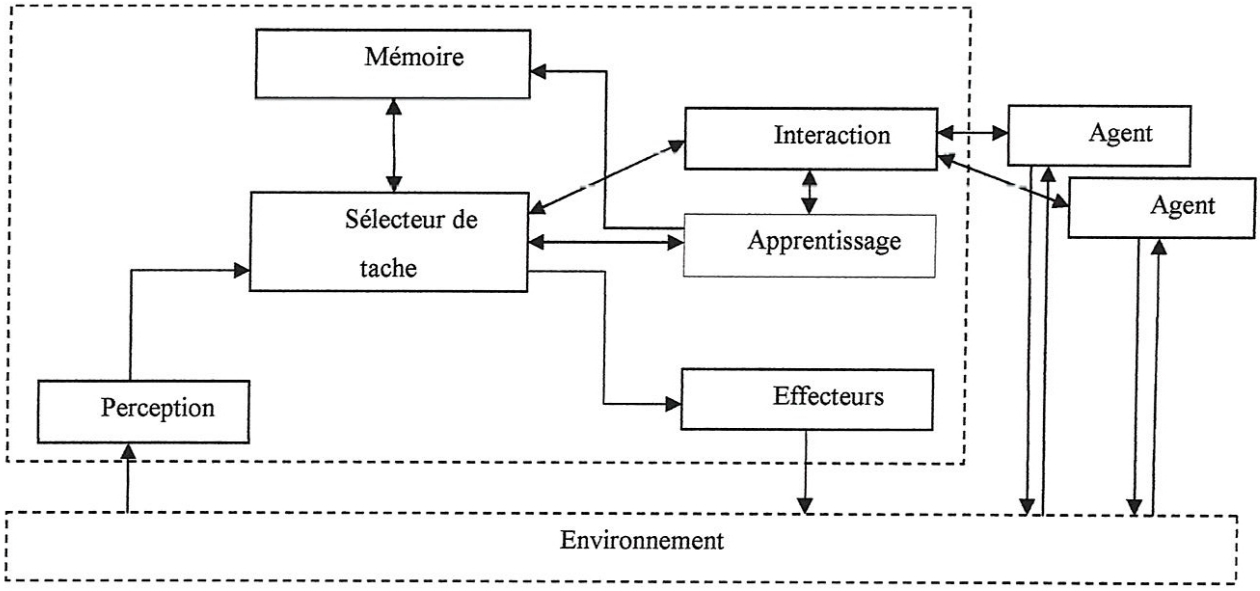

Figure 4.1: Conception d'un agent réactif à base du formalisme Interac-DEC-POMDP

Un agent selon notre modèle de conception, est composé de six modules principaux en relation entre eux, pour accomplir un comportement spécifique, c'est-à-dire que chaque module à besoin d'une information ou d'un échange avec un autre module, pour arriver a choisir le comportement adéquat selon les perceptions, le comportement courant de l'agent et le contenu de la mémoire (les trois Q-valeurs).

L'exécution du système passe par deux phases essentielles : la première est une phase d'apprentissage dont l'agent doit choisir l'un de ses comportements (spécifiés ultérieurement) ; à la sortie de cette phase, les trois politiques d'action, de déclenchement et de résolution d'interaction sont ainsi obtenus et stockées dans la mémoire. Ces politiques sont atteintes en utilisant un processus d'apprentissage qui a été décrit dans la section précédente (section 4.1). La deuxième phase est une phase d'exploitation (ou bien de déroulement de la tache) ; à cet instant les agents vont choisir leurs comportements en se basant sur les trois politiques résultant de la phase d'apprentissage contenues dans des tables.

Description des différents modules :

1. *Module mémoire :* espace de stockage où les trois types de Q-valeurs d'un agent (chaque politique est représentée par une table de Q-valeurs) sont stockés durant l'apprentissage, et utilisés durant l'exploitation du système. Comme nous souhaitons disposer des agents

les plus simples, qui ont la possibilité de s'adapter automatiquement, nous avons décidé de nous concentrer sur des agents avec mémoire à long terme, pour garder une trace de leurs passés.

2. *Module de perception :* il est chargé de restituer les différentes observations de l'environnement. Ces observations ont un rôle très intéressant pour décider quel comportement choisir. Plusieurs types de perception sont possibles: soit c'est une base (endroit où l'agent doit déposer le minerai collecté) ; soit c'est une source (endroit dispersé aléatoirement dans l'environnement, chaque source contenant un nombre limité de minerais ; elle à deux états, pleine, ou épuisée) ; soit un agent (chargé de collecter le minerai tout en évitant les autres agents), soit c'est un obstacle (un obstacle est un carré géométrique, dispersé de façon aléatoire dans l'environnement).

3. *Sélecteur de tâche :* utilisant comme entrées, les perceptions locales et le comportement de l'agent, ainsi que les Q-valeurs contenues dans la mémoire ; le sélecteur de tache choisit entre un ensemble de comportements de l'agent, sachant que le comportement global de l'agent est composé d'un ensemble de comportements élémentaires très simples (réactifs). En appliquant un algorithme de résolution du formalisme utilisé, on peut passer d'un état courant à une sélection d'un nouveau comportement [Vincent, 2005].

4. *Interaction :* les agents que l'on veut concevoir ont la possibilité d'interagir avec les autres agents du monde ; ce module est responsable des interactions possibles qu'un agent peut déclencher en fonction de ses perceptions.

5. *Apprentissage :* Responsable d'amélioration des connaissances (apprentissage) ; c'est à ce niveau que s'effectue l'apprentissage des différentes politiques (d'action, de déclenchement et d'interaction), les améliorations des valeurs étant stockées au niveau de la mémoire.

6. *Effecteurs :* Chargée d'exécuter le comportement choisi par le sélecteur de tâches ; ainsi l'exécution des actions choisies entraine une modification de l'environnement, et va déclencher automatiquement d'autres comportements.

4.3 Choix de l'application : Problème des Robots Fourrageurs

4.3.1 Définition du Problème:

Le problème des robots fourrageurs (notamment robots explorateurs) est apparu dans la communauté de l'IA vers la fin des années 80 (L. Steels [Steels, 1989], R. Brooks [Brook et al, 1990]). Les agents considérés dans ce problème doivent effectuer des tâches de

fourragement. En effet, les robots doivent trouver, récupérer et transporter des échantillons, comme peuvent le faire des insectes avec des brindilles [Drougoul, 1993]. La figure4.1 présente ainsi un système de robots fourrageurs.

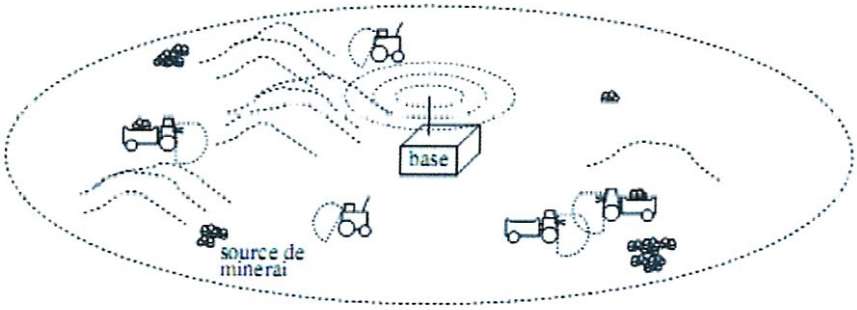

Figure 4.2 : Représentation d'un système de Robots Fourrageurs [Ferber, 1995].

Une définition exhaustive est donnée par Luc Steels [Steels, 1989], et permet une très bonne présentation du problème : *« The objective is to explore a distant planet, more concretely, to collect samples of a particular type of precious rock. The location of the rock samples is not known in advance, but they are typically clustered in certain spots. A number of autonomous vehicles are available that can drive around the planet collecting samples and later reenter the mother ship spacecraft to go back to earth. There is no detailed map of the planet available, although it is known that the terrain is full of obstacles-hills, valleys, etc. Which prevent the vehicles from exchanging any communication. »*[2]

Plusieurs tentatives sont utilisées pour la résolution de ce problème relatif à la nature des robots, doivent-ils être identiques ou hétérogènes. Certains ont proposé de traiter le problème par un groupe hétérogène d'agents (explorateurs, transporteurs et ravitailleurs), d'autres se prêtent bien à la résolution par des systèmes homogènes. Dans notre travail nous traiterons le problème par un ensemble homogène d'agents, c'est-à-dire qui sont pourvus des mêmes capacités.

4.3.2 Description et contraintes :

Les robots fourrageurs qu'on veut réaliser sont en fait des agents qui évoluent dans un environnement (grille) de taille N*M, contenant des obstacles de formes carré ; la grille de

[2] L'objectif est d'explorer une planète distante, plus concrètement, de collecter des échantillons d'un type particulier de roche précieuse. L'emplacement des échantillons de roches n'est pas connu à l'avance, mais ils sont typiquement regroupés en certains lieux. Plusieurs véhicules autonomes sont capables de parcourir la planète en collectant des échantillons et plus tard de retourner au vaisseau mère pour les ramener sur terre. Il n'y a pas de carte détaillée de la planète, toutefois il est connu que le terrain est rempli d'obstacles-collines, vallées, etc. qui empêchent les véhicules d'échanger toute communication.

test contient des cases (espace infranchissable par les agents). Les obstacles et les blocs de minerai et le conteneur (ou coffre servant de réceptacle aux minerais découverts par les agents) sont placés aléatoirement dans la grille.

Les agents fourrageurs que l'on veut concevoir sont limitées par les contraintes suivantes :

1. Chaque agent est désigné par un indice (nombre naturel)
2. Chaque agent ne peut porter que trois blocs de minerai à la fois.
3. Chaque agent possède une benne dans laquelle il va transporter les blocs de minerai.
4. La position du coffre est définie pour chaque agent, alors que la position des blocs de minerai est non définie.
5. Chaque agent est défini par un état individuel ; cet état correspond au couple constitué par sa position (case) et la variable booléenne 'état de sa benne' (rempli ou vide), la position de l'agent est très importante puisque c'est elle qui va conditionner la possibilité d'exécuter certaines interactions.

Les actions qu'un agent peut effectuer sont les suivantes:

1. Il peut se déplacer d'une case à l'autre dans les quatre directions : haut, bas, gauche et droite, avec une connaissance des états des 4 cases qui l'entourent (celle de haut, de bas, de gauche et de droite) et celle sur laquelle il se trouve.
2. S'il est situé sur une case connexe à une source, il peut remplir sa benne.
3. S'il est situé sur une case connexe au coffre et sa benne étant remplie, il peut verser le contenu de la benne dans le coffre.
4. Possibilité d'extraire les blocs de minerai (une benne peut prendre jusqu'à 3 blocs de minerai)
5. Les événements suivants produisent des renforcements positifs : prendre du minerai, le déposer dans le coffre et échanger le contenu de la benne.
6. Les événements qui produisent des renforcements négatifs sont : être dans une zone connexe à une source et ne pas prendre de minerai, soit le minerai est déposé hors coffre, soit l'une des situations suivantes :

Figure 4.3 : Retour de l'agent X lorsqu'il cause différents types de collisions (en donnant les actions de l'autre agent participant à la collision)

Les agents peuvent interagir entre eux, et les seules interactions possibles sont soit des *interactions directes* qui correspondent à des échanges des bennes entre deux agents, soit des *interactions d'accès à une case* particulière.

Pour qu'une interaction directe puisse avoir lieu, les deux agents doivent se trouver sur des cases connexes ; la faisabilité de cette interaction est assurée par la fonction booléenne 'possible', qui correspond à la réponse du système à un déclenchement d'interaction ; une interaction directe peut avoir deux résultats :

- Soit l'échange est effectif, le contenu des bennes est échangé et les états individuels des agents impliqués sont modifiés (modification des états des bennes correspondants).
- Soit l'échange est refusé et les états individuels des agents restent inchangés.

Une interaction d'accès aura lieu quand deux agents veulent accéder à la même case (case vide ou source) ; l'accès est donné à l'agent qui à un grand intérêt pour accéder à cette case (cet intérêt est évalué par les Q-valeurs d'interactions apprises), et les états des agents sont modifiés. Ainsi, cette interaction n'est liée à aucune récompense. Son utilisation ne permette pas de résoudre directement la tâche, mais peuvent néanmoins conduire d'autres agents à le faire.

4.4 Comportement des Robots fourrageurs:

Les comportements sont plus généraux que les actions, car ils ne détaillent pas les états. Le répertoire des comportements de base utile pour le fourragement est fixé comme suit :

- Marcher aléatoirement.
- Aller vers source.
- Remplir benne.
- Vider benne.
- Rentrer à la base.
- Échanger de benne.

La combinaison de ces comportements de base peut être intéressante pour résoudre une grande variété de situations. Apprendre à sélectionner des comportements est par définition un problème d'apprentissage par renforcement [Mataric, 1994], car il est fondé sur une corrélation entre les comportements effectués et les résultats reçus en retour. Les algorithmes d'apprentissage tentent de maximiser les récompenses et de minimiser les punitions au cours du temps. L'algorithme d'apprentissage utilisé est celui décrit dans [Vincent, 2005], qui se base sur l'algorithme Q-Learning ; les agents apprennent par eux-mêmes, le mécanisme

d'apprentissage est identique et distribué dans chacun des robots. Chacun de ces comportements est traduit par une sélection d'action, d'interaction ou d'une résolution d'interaction selon les politiques apprises (en phase d'apprentissage). Le comportement désiré d'un agent fourrageurs est de ramener le plus de minerai au coffre, tout en évitant les autres agents. Cette tâche complexe est peut-être décomposée en plusieurs composants réactifs. Ce comportement est traduit par le schéma suivant (figure 4.4) :

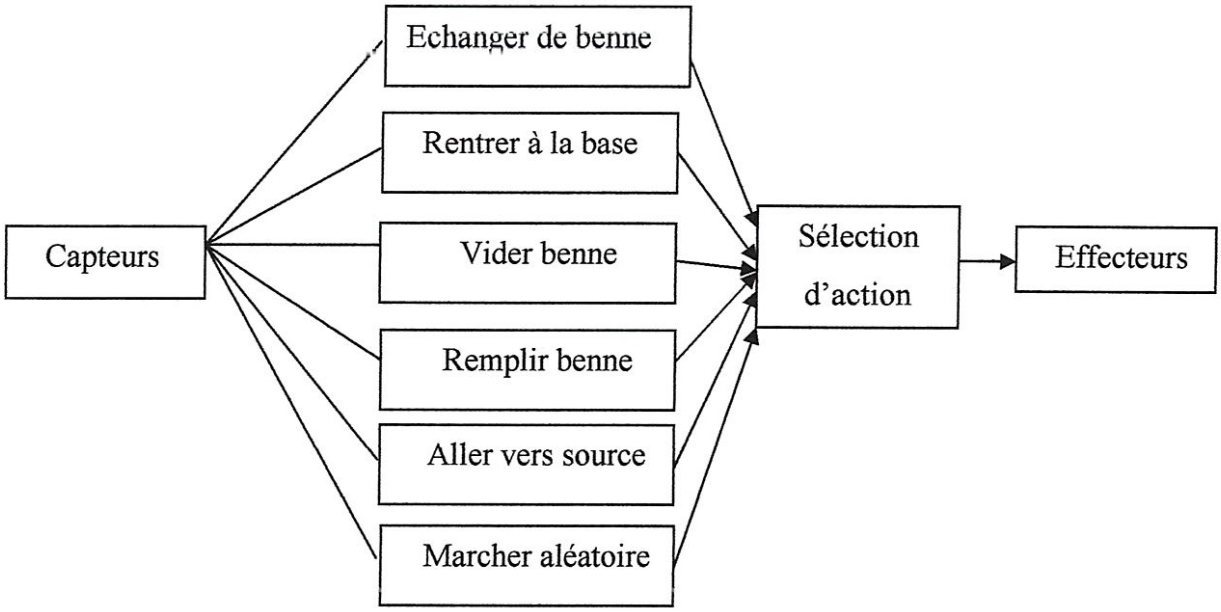

Figure 4.4 : Approche comportementale d'un agent fourrageur

Dans ce schéma, le comportement global est défini par la combinaison d'un ensemble de comportements simples, ces comportements sont ainsi sélectionnés selon les politiques d'action, d'interaction et de déclenchements appris dans la phase d'apprentissage.

4.5 Conclusion:

Dans ce chapitre nous avons expliqué notre modèle de conception basé Interac-DEC-POMDP, ainsi que l'algorithme d'apprentissage utilisé et ses composants ; puis on a expliqué l'application sur laquelle on va tester cet algorithme ; relative à l'application des robots fourrageurs ; suit une description de l'application ainsi que des contraintes pour la modélisation de l'application. On a terminé par l'explication de l'architecture comportementale de nos robots, dont la sélection d'un comportement se fait selon les politiques apprises dans la phase d'apprentissage.

Chapitre V

Expérimentations & Résultats

Chapitre V

Expérimentations & résultats

Dans le chapitre précédent, nous avons présenté le principe de notre modèle de conception d'agent réactif intelligent, à base du formalisme Interac-DEC-POMDP, et les différents modules intervenants. Afin d'évaluer ce modèle de conception, plusieurs expériences ont été faites. Une expérience consiste à définir une instance de problème constitué de plusieurs objets (agents, coffre, sources de minerai et obstacles), les agents ayant des capacités d'interagir directement entre eux. Nous allons détailler les résultats qu'il est possible d'obtenir par l'algorithme d'apprentissage choisi, en mettant l'accent sur un certain nombre de points. Pour que cette approche soit intéressante, elle doit donner de bons résultats concernant le temps de réponse moyen des agents, pour accomplir ensemble la tache de fourragement.

5.1 Présentation du problème :

La modélisation du modèle de conception et sa simulation ont été faites avec le langage de programmation java ; on utilise ainsi des interfaces graphiques pour faire des simulations, en faisant varier un certain nombre de paramètres ; cela ne concerne pas les paramètres d'apprentissage, qui seront fixés dès le début ; les paramètres à prendre en considération sont les suivants :

1. Le nombre d'agents : quel est le nombre nécessaire d'agents pour accomplir la tâche : est-elle accomplie ou non, et esqu'il apparue des problèmes d'accès à la source ou au coffre, ou encore des problèmes de navigations (gènes et collisions).
2. La complexité de l'environnement : Dépend du nombre d'obstacles dispersés dans l'environnement ; ainsi tout en augmentant le nombre d'obstacles, l'environnement devient plus complexe.
3. La position de la base et des sources de minerai : Sont-elles prêts (et donc il ya recours à un rapide accomplissement de la tâche) ; ou loin (il ya plutôt plus d'exploration et donc problèmes de collisions et d'accès à la source).
4. La dimension de l'environnement : La taille est-elle grande ou petite ?

Avant de commencer, il s'avère nécessaire de donner quelques détails complémentaires sur les paramètres d'apprentissages utilisés, à savoir le coefficient d'apprentissage, le coefficient d'actualisation γ, le taux d'exploration ϵ et le renforcement r :

1. Le coefficient d'apprentissage α : Influence le comportement collectif du système, pour cela on le fixe à 0,02.
2. Le coefficient d'actualisation γ : égal à 0,3, il renforce le poids de la tache en cours pour créer une légère persistance ; cette valeur est choisie suffisamment petite pour ne pas empêcher toute fonction d'être déclenché, lorsque l'agent est en exploration.
3. Taux d'exploration ϵ: variante entre 1 et 0.
4. Renforcement r : on affecte une valeur positive à r pour une récompense, et une valeur négative pour une pénalité (déjà fixés dans le chapitre précédent), ou nulle dans les autres cases.

5.2 Étude comportementale des agents :

Avant de commencer une simulation, un certain nombre de paramètres doit être fixé :

1. Fixer le nombre d'obstacles dans l'environnement ; ces obstacles ont tous la même forme (carré) ; les positions des obstacles sont faites de façon aléatoire, et il n'ya pas de possibilité de fixer les obstacles à des endroits bien spécifiques.
2. Fixer le nombre d'agents : Possibilité d'ajouter d'autres agents au cours de la simulation ; les agents sont définis par leurs positions (ligne, colonne) dans l'environnement, par une variable booléenne indiquant le statut de leurs bennes.
3. Fixer le nombre de sources de minerais, sachant qu'il n'ya pas de possibilité d'ajouter d'autres sources (au cours de simulation). Une source est définie par sa position (ligne, colonne) dans l'environnement, par le nombre de blocs de minerai qu'elle contient, et par une variable booléenne indiquant si elle contient encore du minerai ou épuisée.
4. Fixer la position du coffre ; ce dernier est défini par le nombre de blocs de minerai qu'il contient.

Après la fixation de ces paramètres, on peut lancer soit un apprentissage pour apprendre les trois Q-valeurs (d'action, de déclenchement et de résolution d'interaction), soit on lance une simulation (résolution de tache) ; l'arrêt de la simulation est effectuée par le système si le nombre de minerais dans l'environnement est nul.

Le modèle a été testé sur différents environnements, et les évaluations se font à la base de deux critères intéressants : le temps moyen de résolution et le pourcentage passé pour une exploration :

1. Temps moyen de résolution : C'est le temps consommé par l'ensemble des agents pour achever la tache de fourragement, dans une certaine unité de temps, sachant qu'une unité de temps est égale à une itération (cycle de l'algorithme de résolution).
2. Pourcentage du temps dans l'exploration : désigne le pourcentage du temps global de la résolution que passent les agents à la marche aléatoire, en vue de chercher les sources de minerai, son utilité c'est de vérifier que les agents apprennent au mieux leurs politiques d'action.

Différentes simulations effectuées :

Simulation 1 : Cas d'environnement sans obstacle

En se basant toujours sur les mêmes paramètres d'apprentissage fixés auparavant, les paramètres de l'environnement sans obstacle sont choisis comme suit :

- $Dimension = 10 \times 10$
- Une base au centre et quatre sources de minerai.

Cet environnement est représenté en figure 5.1 :

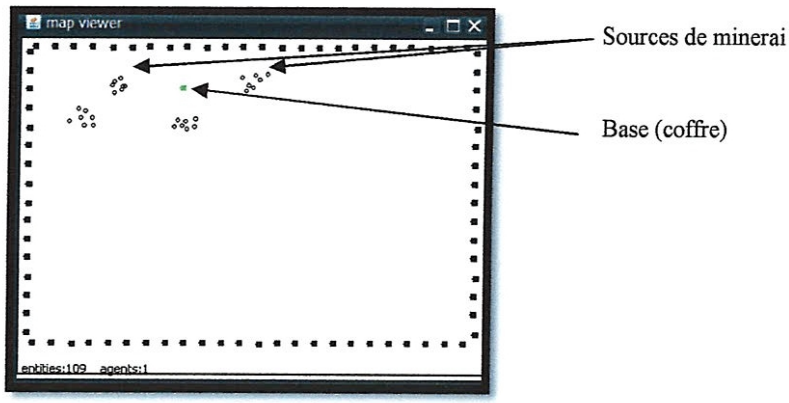

Figure 5.1 : Environnement de simulation 1

En faisant nos simulations sur un total d'agents qui varie entre 3, 10, 15 et 25, on obtient ainsi les résultats contenus dans le tableau 5.1 suivant :

Nombre d'agents	3 agents	10 agents	15 agents	25 agents
Temps moyen de résolution	7800	5133	2003	1024
Pourcentage de temps en exploration (%)	62 %	62,5 %	68 %	65 %

Tableau 5.1 : Tableau des résultats de la simulation 1

Première constatation, le temps de résolution décroit lorsque le nombre des agents augmente ; cela est expliqué par une augmentation de la probabilité de découverte des sources, lorsque le nombre des agents augmente. Pour le pourcentage de temps d'exploration, on voit qu'il représente la plus choisie parmi les différentes politiques existantes (différents comportements), c'est-à-dire que les agents passent la plupart du temps à explorer ou à chercher les sources.

Simulation 2 : Cas d'environnement avec obstacles et sans interaction directe

On continue nos simulations toujours avec un nombre d'agents qui varie entre 3 et 25 agents, et les paramètres de l'environnement sont :

- $Dimension = 10 \times 10$
- Nombre d'obstacles = 20 obstacles
- Une base au centre et quatre sources de minerai (contenant chacune un nombre de minerais similaires où le nombre de blocs est égal à 7).

Cet environnement est représenté en figure 5.2, sa définition répond aux deux critères:

1. Il est dense d'obstacles, pour tester le principe de navigation des robots.
2. Il comporte plusieurs sources, proches ou éloignées de la base, dont les blocs de minerai sont égaux à 7 par source.

On obtient ainsi les résultats présentés dans le tableau 5.2, contenant le temps moyen de résolution et le temps consommé en exploration :

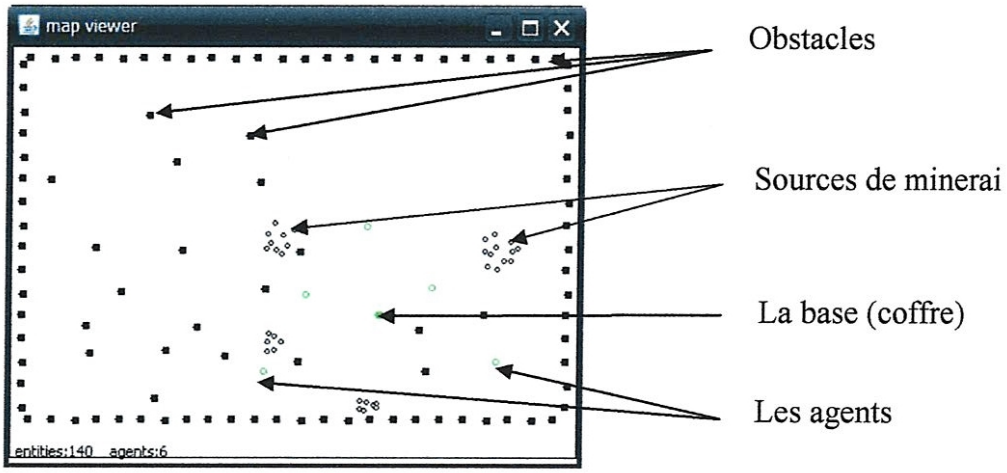

Figure 5.2 : Environnement de simulation 2

Modèle de conception	3 agents	10 agents	15 agents	25 agents
Temps moyen de résolution	9900	5770	3340	2500
Pourcentage de temps en exploration (%)	90,5 %	91%	90 %	91 %

Tableau 5.2 : Tableau des résultats de la simulation 2

Discussion sur les résultats :

D'après les résultats de simulation obtenus, on constate que le temps de résolution décroit lorsque le nombre d'agents augmente. Sur l'espace de 3 à 25 agents, nous pouvons distinguer que :

- Le passage de 3 à 10 agents : Le temps moyen de résolution diminué d'un pourcentage simple.
- Le passage de 10 à 15 : Il ya diminution de temps moyen de résolution ; cette diminution reste significative puisque le nombre des agents n'est pas encore grand, mais le temps passé en exploration au contraire il augmente, ce qui prouve l'absence de coordination concernant l'exploration de l'environnement, et que les espaces sont visités pas mal de fois par plus d'agents, de même que l'agent peut visiter un état plusieurs fois.
- Le passage de 15 à 25 : le temps de réponse moyen décroit à une valeur suffisante, mais il y a une augmentation du temps d'exploration.

Si on regarde bien les résultats, on déduit que le temps passé à l'exploration est très coûteux par rapport aux autres fonctions (comportement de l'agent), mais le temps de résolution reste toujours appréciable si on augmente le nombre d'agents impliqués dans la résolution de la tâche. Cette augmentation est due à la structure de l'environnement, premièrement il n'y a pas trop de sources de minerai, de plus, si elles sont près de la base, le nombre d'obstacles est petit et dispersé de manière à faciliter le passage des agents ; on essayera dans la simulation suivante de compliquer la vue en modifiant l'environnement.

En comparant ces résultats avec ceux obtenus dans la simulation précédente, on conclut que la présence d'obstacles dans l'environnement influence le temps de résolution ; ce dernier a augmenté d'une façon significative sachant qu'il n'y a pas d'interaction directe entre les agents, même le temps d'exploration a augmenté de façon aussi significative.

Simulation 3 : Cas d'environnement avec obstacles et sans interaction directe

On continue la même simulation précédente avec modification de certains paramètres comme suit :

- *Dimension* = 10×10
- 40 obstacles.
- Une base au centre et 8 sources de minerai (contenant chacune 7 blocs de minerai).

Cet environnement est représenté en figure 5.3, les résultats sont donnés par le tableau 5.3 :

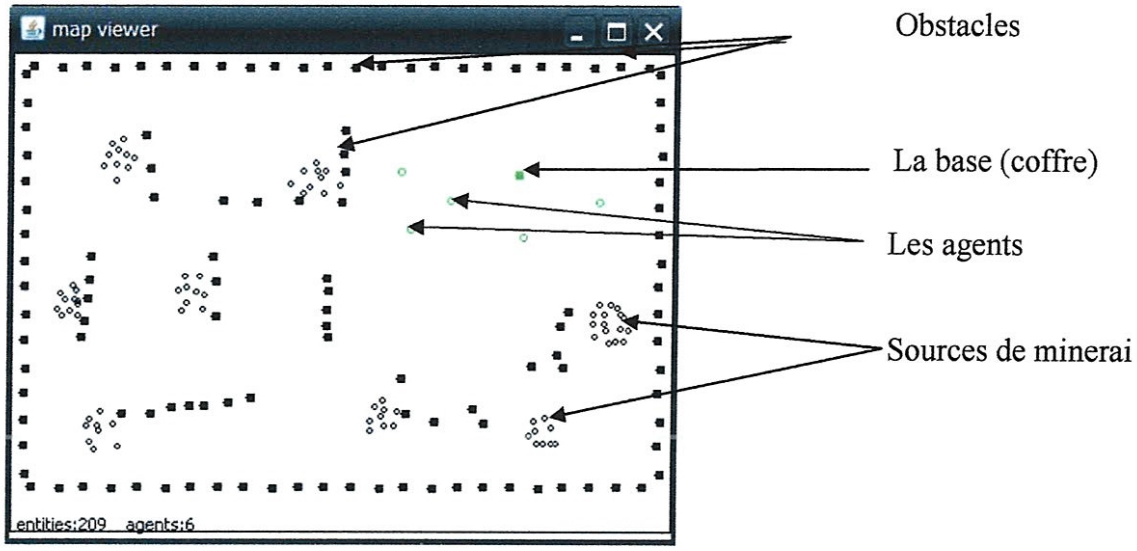

Figure 5.3 : Environnement de simulation 3

Nombre d'agents	3 agents	10 agents	15 agents	25 agents	30 agents
Temps moyen de résolution	30000	10000	3500	3240.5	3700
Pourcentage de temps en exploration (%)	92,5 %	92%	91 %	93 %	93,8%

Tableau 5.3 : Tableau des résultats de la simulation 3

Discussion sur les résultats :

D'après le tableau on peut dire que le temps de résolution moyen décroit lorsqu'on augmente le nombre d'agents (même résultat que le tableau suivant), Sur l'espace de 3 à 30 agents, nous pouvons voir que :

- Le passage de 3 à 10 agents : le temps moyen de résolution diminué de plus de moitié.

- Le passage de 10 à 15 : diminution de temps moyen de résolution ; cette diminution reste significative puisque le nombre d'agents n'est pas encore grand, et le temps d'exploration à aussi diminué, ce qui veut dire qu'il y a une bonne sélection des politiques.
- Le passage de 15 à 30 : le temps de réponse moyen décroit à une valeur suffisante, mais il y a une augmentation du temps d'exploration.

En comparant les deux tableaux précédents, on peut dire que le temps de résolution a augmenté dans la deuxième simulation ; cela est dû à la forme de l'environnement complexe (plus d'obstacles) ; le temps d'exploration a aussi augmenté, ce qui veut dire que les agents ont dans la plupart du temps recours à une exploration, et les politiques ne sont pas bien sélectionnées.

Dans les deux simulations précédentes : lorsque le nombre d'agents est faible (<= 10) chaque agent participe efficacement à la résolution ; si le nombre d'agents est grand (entre 15 et 30), la probabilité de redondance d'exploitation est très forte ; donc l'ajout d'un agent au système n'améliore que la façon linéaire des performances ; ceci est dû à l'absence de l'interaction entre les agents, c'est-à-dire qu'ils ne peuvent pas échanger de minerai entre eux (pas de coopération), et ne peuvent pas exploiter une interaction pour un accès à une case ; cette implémentation ressemble à une architecture d'agent classique purement réactif.

Toutes les mesures de performances ont été accompagnées de l'évaluation d'un indice de temps passé par les agents en tâche d'exploration (ou marche aléatoire) ; quel que soit le nombre d'agents, la moyenne du temps passé dans cette fonction, par tous les agents, est proche de 90 % de la durée totale ; cette valeur reflète l'inefficacité de la stratégie testée, qui est due à l'exploration par marche aléatoire, et à l'impossibilité des agents d'échanger des bennes ou encore les informations. Dans ce qui suit, on va suivre une simulation se basant sur l'échange de minerai entre les agents, et sur des interactions directes pour accéder à une source, un coffre ou une case.

Simulation 4 : Cas d'environnement avec obstacles et avec interaction directe

En reprenant la simulation 3, et en autorisant les interactions entre les agents, on obtient ainsi les résultats représentés dans le tableau 5.4.

Paramètres de l'environnement :

- $Dimension = 10 \times 10$
- 40 obstacles.

- Une base au centre et 8 sources de minerai (avec chacune contient 7 blocs de minerai).

Nombre des agents	3 agents	10 agents	25 agents	30 agents
Temps moyen de résolution	8610	6710	3066	2180
Pourcentage de temps en exploration (%)	90 %	89,5%	85,7 %	85,88 %

Tableau 5.4 : Tableau des résultats de la simulation 4

D'après les résultats du tableau, l'amélioration des performances est faible, mais croit avec le nombre d'agents, donnant un bon temps de résolution par rapport aux deux simulations précédentes. Avec un nombre d'agents égal à 30, et après une période de temps du début de la simulation, on voit que les agents commencent à former des chaines entre les sources et la base (cela est dû aux échanges des bennes entre les agents), mais les chaines sont petites par rapport au nombre des sources de minerai (dû spécifiquement au manque de coordination réactive entre les agents ; retenons que les seules interactions dans notre modèle sont soit des échanges de bennes, soit des interactions d'accès à une case), la figure 5.4 montre ce phénomène

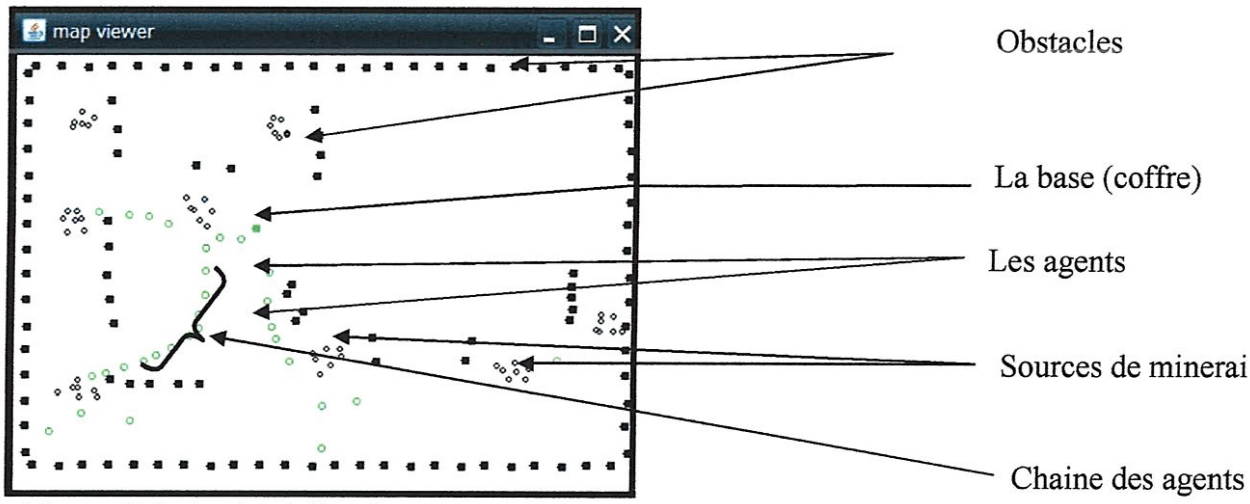

Figure 5.4 : la simulation 4

Les résultats obtenus dans cette troisième simulation sont aussi satisfaisants par rapport aux résultats des deux simulations du modèle sans interaction (simulation 2, simulation 3), mais qui reste à développer ; il peut y avoir plus d'amélioration dans le temps de résolution si on ajoute des techniques de coordination réactive dans notre résolution. Le but c'est de diminuer le nombre d'explorations par les agents, et de faire passer par l'environnement les zones pertinentes (les sources de minerai).

5.3 Influence du coefficient d'apprentissage :

Enfin, nous nous sommes intéressés à l'influence du coefficient d'apprentissage α, sur le comportement collectif construit. Pour cela, nous avons effectué les mêmes expériences que les précédentes, en utilisant un coefficient d'apprentissage α égal à 0,2 (au lieu de 0,02). Dans ce cas, des phénomènes d'oscillations apparaissent, provenant du fait que les agents apprennent trop rapidement.

5.4 Positionnement du travail fait :

Maintenant que nous avons décrit et analysé les résultats obtenus, il est possible de situer notre travail à deux travaux principaux :

5.4.1 Le modèle satisfaction altruisme :

Le modèle satisfaction altruisme [Simonin, 2001] est très proche de notre travail ; il est fondé sur des interactions directes et des interactions indirectes entre les agents utilise des comportements donnés à priori, permettant d'intégrer les signaux émis par les agents au niveau des comportements individuels.

Notre approche utilise des interactions directes et indirectes mais pas de comportements définis à priori ; nous effectuerons un apprentissage automatique, des interactions, ce qui permet de produire et d'adapter les comportements des agents à partir d'un problème donné a priori. Les résultats obtenus dans les simulations faites sont proches des résultats des travaux exposés dans [Simonin, 2001], ce qui ne permet de juger de la satisfaction de nos résultats.

5.4.2 Le formalisme Interac-DEC-POMDP :

Les travaux de Vincent Thomas sont très proches de notre travail ; en fait ils nous ont servi de source d'inspiration, on a utilisé ainsi le formalisme et les algorithmes décrits dans cette thèse pour représenter et résoudre le problème des robots fourrageurs en faisant référence toujours aux heuristiques donnés dans [Vincent, 2005] ; dans ce dernier l'approche est testée sur une application du pompier.

5.5 Conclusion :

Dans ce chapitre, nous avons présenté les différents tests faits. En faisant varier les paramètres décrits au début du chapitre, on a obtenu un ensemble de résultats que l'on peut juger intéressant.

On peut avouer qu'il est possible d'améliorer beaucoup plus nos résultats, en intégrant une technique de coordination réactive à notre approche de résolution. Cette technique de coordination diminuera le temps passé en exploration, en signalant aux agents les zones pertinentes (une fois qu'une source est détectée, l'agent qui est à l' origine de ceci, va tracer par exemple un chemin attractif de la source vers la base, que chaque agent va le suivre vers la source en éliminant ainsi une phase d'exploration).

Partie B

Mise en oeuvre

Conclusion

Dans cette partie, nous avons commencé par exposer l'algorithme d'apprentissage utilisé pour apprendre les comportements des agents, puis on a expliqué le modèle de l'agent et les différents modules le composant, enfin, on a expliqué le comportement global de l'agent comme l'agencement de plusieurs comportements simple et élémentaire ; ces comportements sont sélectionnés à l'aide des Q-valeurs d'actions apprises lors de la phase d'apprentissage.

On a terminé la partie par la présentation des différentes simulations faites ainsi que les résultats obtenus.

Conclusion Générale et Perspectives

Conclusion Générale et perspectives

Lorsque plusieurs agents travaillent sur le même lieu, utilisent les même ressources et résolvent des sous problèmes qui ne sont pas complètement indépendants (conception d'un objet complexe par exemple), ils doivent coordonner leurs actions pour améliorer le fonctionnement du système ; donc ces agents doivent accomplir, en plus des tâches liées directement au problème traité, des tâches de coordination qui ne sont pas directement productives mais qui améliorent le fonctionnement des tâches productives. Plusieurs approches de résolutions ont vu le jour ; certaines se basant sur l'utilisation d'un coordinateur central, d'autres sur la résolution des problèmes de coordinations par les agents eux-mêmes. Les populations animales et celle humaines forment deux sources d'inspiration très intéressantes, permettant de proposer de nouvelles approches de résolutions.

Pour la réduction du rôle du concepteur dans le processus de construction du système, et la concentration sur des constructions automatiques, on s'oriente vers l'utilisation de cadre formel pour exprimer le problème, la caractérisation des comportements des agents, et les manipuler pour produire une réponse collective à ce problème.

Le travail exposé dans ce manuscrit décrit une approche de résolution des problèmes de coordination ; cette approche combine entre la modélisation du système par Interac-DEC-POMDP et la création des comportements par apprentissage par renforcement. Ce dernier offre une structure permettant de prendre des décisions collectives locales ; une des caractéristiques du formalisme réside dans le fait que les actions et les interactions sont forcément dues à des initiatives individuelles : une action est décidée par un agent et une interaction est déclenchée par un autre, aussi l'introduction de la notion d'interaction directe, permet à des agents de raisonner explicitement sur la présence d'autres agents dans le système, et de pouvoir prendre en compte les caractéristiques comportementales des autres agents avec lesquels il est possible d'interagir. On a utilisé un processus d'apprentissage permettant de tirer parti de la notion d'interaction, pour construire automatiquement des systèmes fournissant une réponse collective à des problèmes complexes pour lesquels :

- Les agents sont limités dans leurs interactions
- Les agents ne disposent que d'observations partielles

Le processus d'apprentissage utilisé est défini dans [Vincent, 2005] est constitué par :

1. Des apprentissages par renforcement individuels décentralisés
2. Des échanges de fonctions de Q-valeurs pour prendre des décisions collectives locales, à partir des apprentissages individuels
3. Des échanges de récompenses sociales permettant de distribuer la tache au sein de la collectivité.

Les agents concernés par notre travail sont des robots fourrageurs chargés de collecter une sorte de minerai ; ce problème est traité par plusieurs travaux, et plusieurs solutions ont donné de bons résultats utilisant des inspirations biologiques des populations animales, soit des techniques d'apprentissages. Notre solution consiste à utiliser un formalisme pour la modélisation du problème (Interac-DEC-POMDP) un algorithme d'exploitation pour la résolution de ce formalisme ainsi qu'un algorithme d'apprentissage pour garantir l'adaptabilité de nos agents.

On a proposé un modèle (architecture interne) d'agent fourrageur, et une architecture comportementale où les agents ont aussi la possibilité de coopérer en utilisant un échange de benne entre eux. Les expériences que nous avons effectuées ont ainsi montré l'efficacité de notre travail dans ce domaine, et la possibilité de construire des comportements collectifs à partir de comportements individuels des agents.

Perspectives :

Notre travail s'est inspiré de celui de Vincent Thomas [Vincent, 2005] qui a testé ses propositions sur une application des pompiers, en passant par l'une de ses perspectives, qui est l'utilisation du formalisme et des heuristiques proposés dans d'autres application ; en développant cette perspective et en appliquant ses différents outils à une application des robots fourrageurs, les simulations ont ainsi prouvé l'efficacité des propositions dans le cadre de l'application des robots fourrageurs. A notre tour, nous avons plusieurs points, dont certains peuvent êtres caractérisés dans un proche avenir, d'autres dans un futur lointain.

Perspectives à court terme : Ces points peuvent êtres réalisés dans les meilleurs délais et sont :

1. Utiliser d'autres techniques d'apprentissage.
2. Etendre la notion d'interaction à plus de deux agents.
3. Au lieu d'utiliser des SMA réactifs centré sur les agents (ACMAS), on veut utiliser des SMA orientés sur l'organisation (OCMAS).

4. Au lieu d'utiliser un langage de programmation, où l'exploitation de ces paradigmes pour la notion d'agents est difficile, on peut passer à l'utilisation d'une plateforme dédiée comme Madkit, Swarm ou encore starlogo.

Perspectives à long terme : ces points peuvent être réalisées dans le futur lointain en étudiant forcément d'autres outils qui nous aideront à les réaliser :

1. Proposer un méta modèle pour la construction du SMA dans sa généralité, sans être spécialisé à une sorte particulière des applications des SMA.

Bibliographie

BIBLIOGRAPHIE

A

[Allen et Penault, 1980] J. F. Allen et C. R. Perrault: Analyzing intention in dialogues. *Artificial Intelligence,* 1980.

[Anne, 2002] Anne Nicole : Les systèmes multi-agents, *cours informatique,* 2002

[Astrom, 1965] Astrom, K.J: Optimal control of Markov Decision Processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 1965

B

[Babin et al, 1997] G. Babin, Z. Maamar et B. Chaib-draa: Metadatabase meets distributed AI. *In First Interational Workshop on Cooperative Information Agents (CIA-97),* 1997

[Bellman, 1957] Bellman R.: Dynamic Programming. *Princeton University Press, Princeton, N.J.,* 1957

[Bernestrein et al, 2002] Bernstrein, D., Zilberstein, S. et Immerman, N: The complexity of decentralized control of MDPs. *In Mathematics of Operations Research,* 2002

[Bernestrein et al, 2005] Bernstrein, D.,Hanssen, E.A. et Zilberstein, S. : Bounded policy iteraction for decentralized POMDPs. *In Proceedings of the Ninteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland,* 2005

[Bertsekas, 1995] Bertsekas, D.P.: Dynamic Programming and Optimal Control, *Athena Scientific, Belmont, MA,* 1995.

[Bertsekas et Tsitsiklis, 1996] D. Bertsekas et J. Tsitsiklis: Neuro-dynamic programming, *Athena Scientific, Belmont, MA,* 1996.

[Beynier, 2006] Aurélie Beynier : une contribution à la résolution des PDM décentralisés avec contraints temporelles, *Thèse de doctorat, Université de Caen Basse-Normandie*, 2006

[Bond et Gasser, 1988] A. H. Bond et L. Gasser, editors. Readings in Distributed Artificial Intelligence. *Morgan Kaufmann Publishers: San Mateo, CA,* 1988.

[Bousquet et Lepage, 2001] F. Bousquet et C. Lepage : Systèmes multi-agents et écosystèmes. *Le présent volume*, 2001

[Bratman, 1987] M. Bratman. Intention, plans, and practical reason. *Harvard University Press,* 1987.

[Briot et Demazeau, 2001] Briot J.P et Demazeau Y : Principes et architecture des systèmes multi-agents. *Collection IC2. Hermes-lavoisier,* 2001.

[Brooks et al, 1990] R.brooks, P.Maes, P.Mataric et G.More: Lunar base construction robots, *IEEE international workshop intelligent robots and systems,* 1990.

[Brustoloni, 1991] J. Brustoloni. Autonomous Agents: Characterization and Requirements, *Technical Report CMU-CS-91-204, School of Computer Science, Carnegie Mellon University,* November 1991.

C

[Chaib-draa et Levesque, 1994a] B. Chaib-draa et P. Levesque: Hierarchical models and communication in multi agent environments. In *Proceedings of the Sixth European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-94),* Odense, Denmark, August 1994.

[Chaib-draa, 1994b] B. Chaib-draa: Distributed Artificial Intelligence: An overview. In A. Ken, J. G. Williams, C. M. Hall, and R.Kent, editors, Encyclopedia Of Computer Science And Technology, Marcel Dekker, 1994.

[Maudet et Chaib-draa, 1996] N. Maudet et B.*Chaib-draa*: Commitment-based and Dialogue-game based Protocols--News Trends, *Journal of Experimental and Theoretical AI,* 1996.

[Chaib-Draa, 1999] Chaib Draa B: Agents et systèmes multi-agents. Notes de cours, *Département d'informatique, Faculté des Sciences et de Génie. Université Laval Québec,* Novembre 1999.

[Cohen et Levesque, 1990a] P. R. Cohen et H. J. Levesque: Intention is choice with commitment. *Artificial Intelligence,* 1990.

[Cohen et Levesque, 1990b] P. R. Cohen et H. J. Levesque: Rational interaction as the basis for communication. *In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, Intentions in Communication, The MIT Press: Cambridge,* 1990.

D

[Dayan, 1992] Dayan P.: The convergence of TD(λ) for general λ. Machine Learning, 1992

[Dayan et Sjnowski, 1994] Dayan P. et Sjnowski T.J: TD (λ) converges with probability 1, Machine Learning, 1994.

[Dennett, 1987] D.C. Dennett: The intentional stance, *the MIT press*, 1987.

[Demazeau, 1995] Y.Demazeau: From interaction to collective behavior in agent-based systems, *proceeding of the first eurpean conference on cognitive science, Saint-Malo,* 1995.

[Drake, 1962] Drake, R.: Observation of a Markov Process through a noisy channel, *Phd thesis, Massachusetts Institue of Technology,* 1962

[Drougoul et Ferber, 1992] Drougoul A, Ferber J: Multi-Agent Simulation as a Tool for Modelling Societies: Application to Social Differentiation in Ant Colonies. *Proceedings of Maamaw '92, Rome, C. CastelFranchi and E. Werner (Ed.), North-Holland,* 1992.

[Drougoul, 1993] Alexis Drougoul: De la simulation multi-agents à la résolution collective de problèmes, une étude de l'emergence d'organisation dans les SMA, *thèse de doctorat de paris VI,* 1993.

[Drougoul et al, 1995] Drogoul A, Corbara B, Fresneau D, MANTA (1995): New Experimental Results on the Emergence of Ant Societies, in Artificial Societies: the computer simulation of social life, *Gilbert N. & Conte R. (Eds), UCL Press, London,*1995.

E

[Emery-Montemerlo et al, 2004] Emery-Montemerlo, R., Gordon, G., Schneider, J. Et Thrun, S.: Approximate solutions for partially observable stochastic games with common payoffs, *In Proceedings of the Third Joint Conference on Autonomous Agents and MultiAgent Systems,* 2004

F

[Fagin et al, 1995] R. Fagin, I. Y. Halpern, Y. Moses, and M. Y. Vardi: Reasoning about Knowledge. *the MIT Press: Cambridge, MA,* 1995.

[Ferber, 1995] Jacques Ferber: Les systèmes multi-agents: vers une intelligence collective, *interedition,* 1995.

[Florea et al, 2002] Adina Florea, Daniel Kayser, Stephan Pentiuc: Agents Intelligents, *Politechnica University of Buchares,* 2002

[Franklin et Gasser, 1997] S. Franklin et A. Graesser: Is it an agent, or just a program?: A taxonomy for autonomous agents. *In J. P. Mueller, M. Wooldridge, and N. R. Jennings, editors, Intelligent Agents III: Theories, Architectures, and LanguagesSpringer-Verlag: Heidelberg, Germany,* 1997.

G

[Georgeff et al, 1999] M.Georgeff, B. Pollack, M.Tambe et M.Wooldridge: The belief-desire-intention model of agency, *dans J.P.Muller, M.Singh and A.Rao, Intelligent agents, springer-verlag lecture notes, March* 1999.

[Georgeff et Lansky, 1987] M. P. Georgeff et A. L. Lansky. Reactive reasoning and planning. *In The Proceedings of AAAI-87Seattle,* 1987.

[Gray, 1998] R. Gray : Agent tcl: A flexible and secure mobile-agent systems. *Technical Report PCS-TR98-327, Dartmouth College, Computer Science, Hanover, NH, January* 1998.

[Groupe PDMIA, 2005] Processus Décisionnels de Markov et Intelligence Artificielle, *Groupe PDMIA, version 1.2,* 2005

H

[Hansen et al, 2004] E. A Hansen, D. S Bernstein et S. Zilberstein: Dynamic programing for partially observable stochastic games. *In Proceedins of Nineteenth National Conference on Artificial Intelligence,* 2004

[Hewitt, 1986] Hewitt, C.: Offices are open systems. *ACM Trans. Inf. Syst,* 1986

[Hubbard, 2002] Hubbard J. H. et B. B. Hubbard. Vector calculus, linear algebra, and differential forms-a unified approach. *Prentice Hall,* 2002.

I

[Iglesias et al, 1997] C.A. Iglesias, M. Garijo, J.C. Gonzàlcz, ct J.R. Vclasco: Analysis and design of multi-agent systems using mas-commonkads. *In AAAI'97 Workshop on Agent Theories, Architectures and Languages, Providence,* 1997.

J

[Jennings et Wooldridge, 1998] N.R. Jennings et M.J. Wooldridge: Agent Technology: Foundations, applications, and Markets. *Springer-Verlag: Heidelberg, Germany,* 1998.

[Jennings et al, 1998] N.R Jennings, M.Wooldridge et K.sycara: A road map of reseach and developement, *international journal of autonomous agents and multi-agent systems,* 1998.

K

[Kaelbling et Littman, 1996] L. P. Kaelbling et M. L. Littman. Reinforcement learning : A survey. Journal of Artificial Intelligent Research, may 1996.

[Kaplan, 1998] F. Kaplan: A new approach to class formation in multi-agent simulation of language evolution. *In Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-2), Paris,* 1998.

[Keil et Goldin, 2003] D.Keil et D.Goldin: Modeling indirect interaction in open computational systems, *first international workshop on theory and practice of open computational systems,* 2003.

[Kobsa, 1989] A. Kobsa: User Models in Dialog Systems. *Springer-Verlag: Heidelberg, Germany,* 1989.

L

[Lakemeyer et Levesque, 1999] G. Lakemeyer et H. Levesque: Query evaluation and progression in AOL knowledge bases. *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden,* 1999.

[Lespérance, 1991] Y. Lespérance: A Formal Theory of Indexical Knowledge and Action. *PhD thesis, Toronto,* 1991.

[Lespérance et al, 1994] Y. Lespérance, H. J. Levesque, F. Lin, D. Marcu, R. Reiter, et R. B. Scherl: A logical approch to high-level programming- A progress report. *In Benjamin Kuipers, editor, In Control of the Physical World by Intelligent Systems, Papers from the 1994 AAAI fall Symposium, New Orleans, LA,* 1994.

M

[Malone, 1988] Malone T. W.: What is coordination theory. *In National Science Foundation Coordination Theory Workshop, MIT,* 1988.

[Malville et Bourdon, 1998] E. Malville et F. Bourdon: Task allocation, A group self-design approach. *In Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS- 2), Kyoto, Japan,* 1998.

[Mataric, 1994] Mataric Maja. J : interaction and intelligent behavior. *Phd of philosophy Massachussettes Institue of Technology,* May 1994.

[Michie et Chambers, 1968] Michie, D., Chambers, R. : BOXES : an experiment in adaptive control, *Machine Intelligence,* 1968.

[Moulin et Chaib-draa, 1996] B. Moulin et B. Chaib-draa: An overview of distributed artificial intelligence. *In G. M. P. O'Hare and N. R. Jennings, editors, Foundations of Distributed AI, John Wiley & Sons: Chichester, England,* 1996.

N

[Nair et al, 2003] Nair, R., Tambe, M., Yokoo, M., Marsella, S. Et Pynadath, D.V.: Taming decentralizedPOMDPs: Towards efficient policy computation for multiagent settings. *In Proceedings of the International Joint Conference on Artificial Intelligence,* 2003

[Negroponte, 1995] A. Negroponte: Being Digital. *Hodder and Stoughton,* 1995.

[Nwana et Ndumu, 1999] H.S. Nwana et D. T. Ndumu: Agents of change in future communication systems. *Applied AI Journal,* 1999.

P

[Papadimitriou et Tsitsiklis, 1987] papadimitriou, C.H. et Tsitsiklis, J.N.: The complexity of Markov Decision Processes. *Mathematics of Operations Research,* 1987

[Peshkin et al, 2000] Peshkin, L., Kim, K., Meuleu, N. Et Kaelbling, L.: Learning to cooperate via policy search. *In sexteenth Conference on Uncertainty in Artificial Intelligence,* 2000

[Phillipe, 2007] Phillipe: Apprentissage par renforcement, *notes de cours, GRAPPA,* 2007.

[Potapov et al, 2003] A.B. Potapov, M.K. Ali: Convergence of reinforcement learning algorithms and acceleration of learning . *Phys. Revue,* 2003.

[Puterman, 1994] M.L. Puterman : Markov Decision Processes : Discrète stochastic dynamique programing, *John Wiley and sons*, 1994

R

[Rao et Georgeff, 1991] A. S. Rao et M. P. Georgeff: Modeling rational agents within a BDI- architecture. *In R. Fikes and E. Sandewall, editors, Proceedings of Knowledge Representation and Reasoning, Morgan Kaufmann Publishers: San Mateo,* April 1991.

[Rao et George, 1995] A. S. Rao et M. P. George: BDI-agents : from theory to practice. *In Proceedings of the First Intl. Conference on Multiagent Systems, San Francisco,* 1995.

[Rus, 1996] D. Rus and D. Subramanian: Information retrieval, information structure, and information agents. *Technical Report PCS-TR96-255, Dartmouth College, Computer Science, Hanover, NH,* January 1996.

[Russell et Norvig, 1995] Stuart J Russell et Peter Norvig: Artificial intelligence: Amodern approach, *prentice hall, first édition,* january 1995.

[Russell et Norvig, 2003] S.Russell et P.Norvig: Artificial Intelligence: A modern approach, *prentice Hall series (second édition),* 2003.

S

[Samuel, 1959] Samuel, A : Some studies in Machine Learning using the game of checkers , *IBM journal of research development,* 1959.

[Sekaran et Sen, 1994] Sekaran Mahendra, Sen Sandip: Multi-Agent Learning In Non Cooperative *Domains ECAI 94,* 1994

[Sen et al, 1994] Sen Sandip, Sekaran Mahendra, et Hale Jones: Learning to coordinate without sharing information. *Proceedings of the twelfth national conference on Artificial Intelligence,* August 1994.

[Shoham, 1993] Y.Shoham: Agent-oriented programming, *artificial intelligence,* 1993.

[Smallwood et Sondik, 1973] Smallwodd, R.D. et Sondik, E.J.: The optimal control of partially observable markov decision processes over a finite horizon, *Operation Research,* 1973

[Sian, 1991] Sian Sati Singh: Adaptation Based On Cooperative Learning In Mas. *In Proceedings of the sevcond workshop on Modelling Autonomous Agents in Multi-Agent World Editors Y. Demazeau & J-P Müller Editions North Holland,* 1991.

[Sigaud, 2004] Sigaud, O.: Comportements adaptatifs pour des agents dans des environnements informatiques complexes. *Habilitation à Diriger des Recherches. Université Pierre et Marie Curie, Paris 6.* 2004.

[Simon, 1947] Simon, H.: Administrative Behavior. A study of Decision-making Processes in Administrative organisation, *Free Press,* 1947

[Simonin, 2006] G. Simonin : Agent et Apprentissage : cours SMA, *Master IGIS,* 2006

[Singh, 1994] M. P. Singh: Multi-agent Systems: A Theoretical Framework for Intentions, Know-How,and Communications. *Springer-Verlag: Heidelberg, Germany,* 1994.

[Steels, 1989] L.steels: Coopération between distributed agents through self-organisatio. *In Demazeau, Y et Muller, J, editors, decentralized AI-Proceedings of the first european workshop on modelling autonomous agents in a multi-agent world,* 1989

[Sutton, 1988] Sutton, R.S.: Learning to Predict by the Method of Temporal Différences, *Machine Learning,* 1988.

[Sutton et al, 1999] Sutton, R., Precup, D. Et Singh, S.: Between MDPs and semi MDPs: Aframework of temporal abstraction in reinforcement learning. *Artificial Intelligence,* 1999

[Sutton et Barto, 1998] R. Sutton et A. Barto: introduction to reinforcement learning, MIT Press, 1998

[Szer et al, 2005] Szer, D., Charpillet, F. Et Zilbersetin, S.: MAA*: A heuristic search algorithm for solving decentralized POMDPs. *In Proceedings of 21 st Conference on Uncertainty in Artificial Intelligence (UAI'2005),* 2005

T

[Thomas, 1993] S. R. Thomas: PLACA, an Agent Oriented Programming Language. *PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305,* August 1993.

U

[Uribe et al, 1999] Uribe A.P et Sanchez E. A comparison of reinforcement learning with eligibility traces and integrated learning, planning and reacting. *Concurrent systems engineering series vol. 54 Ios press, Amsterdam*1999

V

[Vincent et al, 2004]: Vincent Thomas, Christine Bourjot, Vincent Chevrier : Formalisme pour la construction automatique d'interaction dans les SMA réactifs, *version étendue, Rapport de recherche, équipe Maia,* 2004

[Vincent, 2005] Vincent Thomas: Proposition d'un formalisme pour la construction automatique d'interaction dans les SMA réactifs, *phd thèse* , 2005.

[Vincent et al, 2006] Vincent Thomas, Christine Bourjot, Vincent Chevrier Heuristique d'apprentissage automatique décentralisé d'interaction dans des systèmes multi-agents réactifs, *JFSMA'06,* 2006

W

[Watkins, 1989] Watkins, C.J.: Learning from Delayed Rewards, *PHD thesis, Cambridge University, Cambridge, England,* 1989.

[Weib, 1993] Weiß Gerhard: Learning To Coordinate Actions In Multi-Agent Systems. *In Proceedings of the International Joint Conference on Artificial Intelligence,* August 1993

[Weib, 1993] Weiß Gerhard: Learning To Coordinate Actions In Multi-Agent Systems. *In Proceedings of the International Joint Conference on Artificial Intelligence,* August 1993

[Weiss, 1999] Weiss. G: Multi-agent systems, a modern approach to distributed artificial intelligence, *MIT Press,* 1999.

[Wooldridge et Jennings, 1995] M.Wooldridge et N.R.Jennings: Intelligent agents: Theory and practice, *the knowledge engineering review*, 1995.

[Wooldridge, 2002] M.Wooldridge: An introduction to multi-agent systems, *John Wiley and Sons,* 2002.

Web bibliographie:

1. A gentle introduction to agents and their applications
   http://www.magma.ca/~mrw/agents/

2. S. Franklin and A. Gasser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Proc. of ATAL'96.
   http://www.msci.memphis.edu/~franklin/AgentProg.html

3. M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The Belief-Desire-Intention Model of Agency. Dans J. P. Muller, M. Singh, and A. Rao (eds), Intelligent Agents V Springer-Verlag Lecture Notes in AI Volume 1365, March 1999.
   http://http://www.csc.liv.ac.uk/~mjw/pubs/atal98b.pdf

4. M. Wooldridge and N. R. Jennings : Intelligent Agents: Theory and Practice. Knowledge Engineering Review, 1995.
   http://www.csc.liv.ac.uk/~mjw/pubs/ker95/ker95-html.html

5. http://www.emse.fr/~boissier/enseignement/sma01/index.html

6. http://magma.imag.fr/publications/

Annexe

Les Composants des Interactions

1. Composants des interactions :

Les principales situations d'interaction peuvent être classées par rapport à trois critères : les objectifs ou intensions des agents, les relations que les agents entretiennent envers les ressources qu'ils possèdent ainsi que les moyens (ou compétences) dont ils disposent pour parvenir à leurs fins.

Buts compatibles ou incompatibles [Ferber, 1995] :

Les agents ont-ils des buts concordants, c'est-à-dire allant dans le même sens ou bien sont contradictoires. Les buts sont incompatibles si la satisfaction de l'un entraine l'insatisfaction de l'autre et vise vers ça.

Définition : *Le but d'un agent A est incompatible avec celui d'un agent B. p, q sont les buts respectifs de A et B.* $p \Longrightarrow \sim q$, *c'est-à-dire que :* $satisfait\ (A,p) \Longrightarrow \sim satisfait(B,q)$

Les ressources [Ferber, 1995] :

On entend par ressource, tous les éléments environnementaux et matériels utiles à la réalisation d'une action. Tout agent à besoin de ressources pour réaliser ses actions ; cette quantité de ressources nécessairement limitée, est a l'origine des conflits ; ces derniers apparaissent essentiellement lorsque plusieurs agents ont besoin des mêmes ressources en même temps et au même endroit (c'est-à-dire lorsque les zones correspondants aux ressources dont ils ont besoin s'intersectés). Ces conflits peuvent être résolus d'une manière ou d'une autre par l'intermédiaire de mécanismes de coordination d'actions et de résolution de conflits (la loi du plus fort, les méthodes de négociation…).

Les techniques de coordination d'actions sont utilisées pour anticiper les conflits à venir, et faire en sorte qu'ils soient gérés avant même qu'ils apparaissent ; les limitations des ressources peuvent êtres causes de conflit évitable dans une certaine mesure, en coordonnant les actions des agents.

Capacités des agents par rapport aux tâches (compétences) [Ferber, 1995]

Est-ce qu'un agent peut réaliser seul une tache, ou bien a-t-il besoin des autres pour parvenir à son but ?

Un agent isolé s'avère capable de faire face à la réalisation de son but, d'autres buts plus complexes, peuvent être effectués par un individu unique, même si leur accomplissement est facilité par la présence d'autres personnes qui s'apportent ainsi mutuellement une aide. Enfin, certains requièrent impérativement les capacités de plusieurs personnes.

2. Types d'interaction :

Les trois composants de l'interaction vont permettre de faire une première typologie des situations d'interaction (présenté ainsi au chapitre 1, section 1.3.2) ; les différentes situations résultantes sont définies comme suit [Ferber, 1995] :

1. *Indépendance :* buts compatibles, ressources suffisantes et compétences suffisantes ; cela résume la simple juxtaposition des actions des agents pris indépendamment, sans qu'il n'y ait d'interaction.
2. *Collaboration simple :* buts compatibles, ressources suffisantes, et compétences insuffisantes ; elle consiste en une simple addition des compétences ne nécessitant pas d'actions supplémentaires de coordination entre les intervenants.
3. *Encombrement :* buts compatibles, ressources insuffisantes, et les compétences insuffisantes, l'encombrement est caractéristique de toutes les situations dans lesquelles les agents se gênent mutuellement dans l'accomplissement de leurs tâches, alors qu'ils n'ont pas besoins les uns des autres. La résolution de ce type de problème fait appel à des techniques spécifiques de coordination d'actions.
4. *Collaboration coordonnée :* buts compatibles, ressources insuffisantes et compétences insuffisantes, elle suppose que les agents doivent coordonner leurs actions pour pouvoir disposer de la synergie de l'ensemble de leurs compétences. Il s'agit donc de la plus complexe des situations de coopération, puisqu'elle ajoute aux problèmes d'allocation de taches, des aspects de coordination dus aux ressources limitées.
5. *Compétition individuelle pure :* buts incompatibles, ressources suffisantes et compétences suffisantes ; quand les buts sont incompatibles, les agents doivent lutter ou négocier pour atteindre leurs buts ; dans la compétition pure les ressources ne sont pas limitées.
6. *Compétition collective pure :* buts incompatibles, ressources suffisantes et compétences insuffisantes ; lorsque les agents n'ont pas de compétences suffisantes, ils doivent se regrouper au sein de coalition pour parvenir à atteindre leurs objectifs.

7. *Conflit individuel pour des ressources :* buts incompatibles, ressources insuffisantes et compétences suffisantes ; lorsque les ressources ne peuvent pas être partagées, on se trouve dans une situation caractéristique de conflits, dont les ressources sont l'enjeu, chacun voulant l'acquérir pour lui seul.
8. *Conflits collectifs pour des ressources :* buts incompatibles, ressources insuffisantes et compétences insuffisantes ; les coalitions luttent les unes contre les autres pour obtenir le monopole d'une ressource, d'un territoire ou d'une position

3. Notion de coopération:

On dira que plusieurs agents coopèrent, ou encore qu'ils sont dans une situation de coopération, si l'une des deux conditions est vérifiée :

1. L'ajout d'un nouvel agent permet d'accroitre différentiellement les performances du groupe
2. L'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels.