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

UNIVERSITE DE GUELMA
FACULTE DES SCIENCES ET DE L'INGÉNIEUR

Mémoire de Magister

Présenté au département d'informatique

Pour l'obtention du diplôme de

Magister en Informatique

Option : Intelligence Artificielle et Imagerie

Par M^me^ BENHAMZA KARIMA

CONCEPTION D'AGENT INTELLGENT AVEC APPRENTISSAGE PAR RENFORCEMENT

JURY

Président	M^me^	H.SERIDI	M.C	Université d'Annaba
Rapporteur	Mr	H. SERIDI	M.C	Université de Guelma
Examinateur	M^me^	H. BELLEILI	M.C	Université d'Annaba
Examinateur	Mr	T. KHADIR	M.C	Université d'Annaba

2008

Résumé

Le problème de la conception automatique d'un système intelligent est au cœur de nombreuses recherches dans le domaine de l'intelligence artificielle située. C'est un enjeu important qui renvoie en particulier à deux grandes problématiques : la représentation de l'environnement au niveau de l'agent, et la prise de décision avec cette représentation.

Dans ce contexte, nous cherchons à élaborer un processus automatique qui permet de créer des entités intelligentes pouvant prendre, individuellement des décisions, dans un environnement inconnu sans modélisation préalable. L'outil proposé est l'apprentissage par renforcement (AR) et l'algorithme implémenté est le Q-learning.

A l'issue d'expérimentations appliquées sur une plate-forme de robotique mobile, la mise en interaction de telle entité avec l'environnement est étudiée. Ce qui a permis, ensuite, de proposer des justifications pour une spécification rigoureuse des différents paramètres d'apprentissage selon le comportement qui en découle.

Mots clés: Agent situé, processus décisionnel de Markov, apprentissage par renforcement, Q-learning, robotique mobile, paramètres d'apprentissages.

Abstract

The problem with an automated design of intelligent system is the heart of much artificial intelligence research. This significant stake has in particular two great problematic: the representation of the agent environment and the decision making of this representation.

In this context, an automated process for creating intelligent entities that can take individual decision is developed in an unknown environment; without prior modelling. The proposed tool is reinforcement learning i.e. (RL) and the implemented algorithm is the Q-learning.

At the end of platform mobile robotic experiments, interaction with the environment developed by such entity is studied. This allows justifications to be suggested for a rigorous specification of different learning parameters, according to results behaviour.

Keywords: Agent, Markov decision process, reinforcement learning, Q-learning, mobile robotics, learning parameters.

الخلاصـــة

إن تصميم نظام ذكي هو في صميم الكثير من الأبحاث في مجال الذكاء الاصطناعي. وهذه مسألة تشير إلى قضيتين رئيسيتين: تمثيل البيئة عند الوكيل، وصنع القرار مع هذا التمثيل.

في هذا السياق، نسعى إلى تكوين عملية آليه تتيح إنشاء كيانات ذكيه قادرة أن تتخذ قرارات فردية، في بيئة مجهولة دون سابق نمذجة، الاداه المقترحة هي "تعزيز التعلم" و الخوارزم المستعمل هو "ك-لورنين".

وفي نهاية التجارب المطبقة على منهاج الروبوتيك الجوال، تم دراسة تفاعل هذا الكيان مع البيئة وسلوك الناشئ عنه و الذي سمح لنا بتقديم مبررات للاختيار مختلف عناصر التعلم وفقا لمعايير السلوك الناتج.

الكلمات الرئيسية: وكيل، قرار عملية ماركوف، وتعزيز التعلم، ك-لورنين، روبوتيك الجوال، عناصر التعلم

Remerciements

Je tiens à remercier toutes les personnes qui m'ont permis de mener à bien ce travail.

Mes remerciements et ma gratitude s'adressent tout d'abord à **Mr SERIDI Hamid,** M.C à l'Université 08 mai 45- Guelma, pour m'avoir accueillie dans son équipe et fait découvrir les techniques de la recherche. Je lui suis reconnaissante pour sa disponibilité et l'intérêt qu'il a porté à mon travail tout au long de l'année.

Je souhaite remercier **Mme H. SERIDI,** MC à l'Université Badji Mokhtar,-Annaba, pour avoir accepter de présider le jury. Qu'elle me permet de lui exprimer ma plus haute considération.

Je remercie également, **Mme H. BELLEILI et Mr T. KHADIR** MC à l'Université Badji Mokhtar- Annaba pour leur investissements en tant qu'examinateurs et pour le temps qu'ils ont consacré à lire et évaluer ce travail.

Un grand merci à **Mr. R. MUNOS** Directeur de recherche INRIA, France pour ces orientations qui m'ont permis d'appréhender le domaine extrêmement vaste qui est l'apprentissage par renforcement.

Je tiens à remercier plus particulièrement **Mr A. BOULARAIS** Etudiant doctorant à l'université Laval, Canada pour son aide et tous les échanges enrichissants que nous avons eus à propos de mon travail.

Enfin, je remercie tout particulièrement ma famille et mes amis qui m'ont encouragée tout au long de mes études. Leur soutien et leur aide sont pour beaucoup dans l'accomplissement de ce travail.

Sommaire

Liste des figures

Liste des tableaux

Glossaire

α	Coefficient d'apprentissage
ε	Taux d'exploration
γ	Coefficient d'actualisation
π	Stratégie de commande
π^*	Stratégie optimale
☺	Agent position finale (apprentissage)
☹	Agent position initiale (non apprentissage)
P	Probabilité de transition
R	Fonction de récompense
r	Récompense scalaire
t	Temps Période d'échantillonnage
a	Action
A	Ensemble d'action
s	Etat courant
S	Ensemble des états
Q	Fonction d'utilité
Q*	Fonction d'utilité optimale
Q_{init}	Valeur initiale de la fonction d'utilité Q

Introduction générale

Introduction générale

Le but initial de l‘intelligence artificielle (IA), domaine né dans les années 50, était de créer des systèmes informatiques qui égaleraient voire devanceraient l'homme dans de nombreuses activités réputées intelligentes comme raisonner, résoudre des problèmes complexes ou naviguer pour accomplir une mission dans un environnement totalement inconnu tout en s’adaptant au changement de ce dernier. Les résultats n'ayant jamais atteint ces espérances, ce domaine a aujourd'hui pour objectif la compréhension et l'adaptation des mécanismes immanents du comportement "intelligent" pour les appliquer aux systèmes informatiques.

Ainsi, l’IA peut être présentée comme la discipline qui tente de construire des "systèmes informatiques intelligents". Si ses buts et ses fondements n’ont pas changé depuis, les moyens et les méthodes utilisés ont largement évolué, surtout dans les dernières années. L’approche initiale s’est appuyée sur la manipulation et la représentation symbolique de la connaissance. L’idée fondamentale derrière cette orientation est la notion d’abstraction, de raisonnement à partir de symbole pour déduire des faits et construire des plans d’action. Mais les limites de l’IA symbolique pour représenter l'information et ses difficultés à s'adapter à la dynamique et à la réalité de l’environnement, ont éveillé l'intérêt des chercheurs pour de nouvelles approches, et en particulier l'approche "agent".

La notion d’agent situé ou isolé est une approche récente pour aborder la modélisation et la résolution de problèmes en utilisant l’idée que l'on peut automatiquement apprendre en interférant avec son environnement : C’est le concept fondamental de l’intelligence artificielle située (IAS). Son avantage est qu’il offre un bon cadre pour la structuration des informations: les données contenues dans l’agent représentent les objectifs visés et celles intégrées dans l’environnement représentent les ressources externes dont dispose ce dernier. Avec cette boucle sensori-motrice, on arrive finalement à intégrer le raisonnement au sein de cette entité.

C’est probablement la première idée qui vient à l'esprit lorsqu'on examine la nature du phénomène d'apprentissage humain. Tout au long de sa vie, l’homme peut mémoriser les conséquences de ses actes, observer la manière avec laquelle réplique l’environnement et utiliser ces informations pour agir au mieux et prendre à tout moment la meilleure décision. L’interaction est une source majeure de cette connaissance, qui permet de redonner, à nous humains, plus d’autonomie, d’efficacité et finalement d’intelligence.

Nous pouvons alors nous interroger sur ce qu'on entend par « agir au mieux » ou «meilleure décision ». Mathématiquement, la recherche de ce qui est «meilleur » ou «optimal», se traduit en terme de rationalité. Ce dernier est présenté par l'optimisation d'une fonction caractérisant les objectifs de l'agent en introduisant la notion de mesure de performance qui permet d'évaluer une décision en terme de succès. La décision maximisant la mesure de performance est celle qui va garantir le plus de succès à l'agent, elle est alors appelée décision rationnelle.

C’est un concept qui est à la base des théories économiques, largement étudié dans plusieurs domaines tels que la sociologie, les sciences de la gestion, les sciences politiques, la psychologie cognitive, ... et qui vient finalement s’intégrer dans le domaine de l’IA située.

A ses débuts, ce domaine était également considéré comme entièrement séparé des disciplines de la recherche opérationnelle telle la théorie du contrôle ou encore de décision. Les travaux reposaient essentiellement sur la logique, les symboles mais pas les nombres. Depuis plus d’une vingtaine d'années, le domaine a progressivement évolué et s'est ouvert aux méthodes numériques et statistiques. Les chercheurs acceptent ces approches qui sont utilisées comme des mécanismes nécessaires à leurs besoins. Les formalismes mathématiques constituent des outils puissants pour répondre aux problèmes de représentation d’entités devant résoudre un problème et proposent en outre des algorithmes permettant la construction automatique du comportement désiré.

Par ailleurs, l’emploi de ces entités intelligentes en vue de la réalisation de tâches dangereuses, difficiles, impossibles ou simplement répétitives pour l'homme, n'est pas un concept nouveau. Les récents progrès réalisés concernant le développement d'agents autonomes ont toutefois permis de concrétiser de tels projets auparavant cantonnés aux légendes et aux récits de science fiction. La robotique est, en outre, un champ d’action privilégié. Dans la mesure où il s’agit de faire effectuer à un robot une tâche habituellement

dévolue à l'homme, le besoin d'introduire l'intelligence s'est accru contraignant ce domaine à explorer des voies de plus en plus variées.

Il est utile, en effet que les robots partagent quelques capacités humaines telles la perception de l'environnement, l'acquisition de connaissances et la prise de décision. Leur comportement doit être fondé sur la notion de « rationalité » pour être plus flexibles, plus autonomes pour être qualifiés d'intelligents. Un autre aspect relativement important est la possession d'un processus permettant d'effectuer une synthèse des expériences passées pour mettre à jour le comportement désiré : les techniques d'apprentissage répondent à cette problématique.

Finalement, en y intégrant toutes ces techniques et approches, l'étude des systèmes intelligents en IA située est devenue un domaine de recherche à part entière.

Domaine de recherche

A partir des aspects qui viennent d'être évoqués (IA située, Conception, Décision, Apprentissage, Robotique), nous pourrons situer notre sujet de recherche à l'intersection des domaines suivants : L'IAS qui se préoccupe de concevoir des systèmes dit « intelligents » fondés sur une boucle sensori-motrice communément appelée « agent - environnement » et la théorie de décision. Ce domaine mathématique qui se base sur les axiomes de probabilité et d'utilité, constitue un cadre permettant d'évaluer les prises de décision dans des environnements inconnus et incertains. Il propose aussi des formalismes pour la représentation d'entités devant résoudre un problème sous forme d'optimisation.

Nous utiliserons les outils combinés de ces domaines à savoir : « agent intelligent », « Processus décisionnel de Markov » et « Apprentissage par renforcement », sur lesquels s'articulera ce mémoire. Nous étudierons aussi la mise en interaction de telles entités avec l'environnement et les comportements qui en découlent. Ce problème constituera le coeur de notre travail.

Problématique

Les deux grandes problématiques de la conception des agents intelligents sont : la représentation de l'environnement au niveau de l'agent, et la prise de décision avec cette représentation. Nous chercherons donc à élaborer un processus automatique qui permet de créer des entités pouvant prendre, individuellement des décisions, de façon à obtenir un comportement rationnel et autonome. L'outil utilisé est l'apprentissage par renforcement (AR), l'algorithme implémenté est le Q-learning et l'application choisie est la robotique

mobile.

Ensuite, nous développerons une étude qui permettra de proposer des justifications expérimentales pour le choix, d'une part de la fonction de renforcement et d'autre part des valeurs initiales des paramètres d'apprentissage et de la table Q. Nous essayerons de découvrir l'influence de ces éléments sur le processus d'apprentissage.

Plan de ce mémoire

Le présent document se compose en trois parties. La première réalise un état de l'art des différents outils utilisés, la seconde introduit l'approche théorique de conception et enfin, la troisième porte sur le modèle de conception proposé en exposant les différents modules intégrés et les différentes expérimentations menées en vue de synthétiser ce travail.

En nous appuyant sur une étude bibliographique, nous présenterons en première partie de ce document, les différents concepts et outils utilisés dans notre travail. Nous introduisons les systèmes basés « agent » qui s'articulent tout d'abord autour de l'analyse de différentes définitions et notions sur lesquels cette nouvelle discipline repose. Nous situerons, ensuite l'intelligence et les critères choisis pour qualifier une entité d'« intelligente ». Nous conclurons par donner une définition propre à notre agent et une approche de conception basée rationalité.

La seconde partie de ce document portera sur les processus décisionnels de Markov lesquels proposent un formalisme mathématique particulièrement adapté à la résolution de problème décisionnel sous incertitude. Ceci conduira ensuite à présenter les différentes méthodes d'apprentissage par renforcement (AR) en décrivant les algorithmes qui s'avèrent les plus représentatifs. Nous discuterons l'utilisation de ces algorithmes en présentant leurs limites. Nous terminerons cette partie par le domaine d'application de l'apprentissage par renforcement.

La dernière partie confirmera notre choix d'algorithme développé pour la conception d'agent intelligent et relatera les différents modules intégrés dans l'implémentation du système proposé. A l'issue de nos expérimentations appliquées sur une plate-forme de robotique mobile qui va permettre de présenter sa certification, nous présenterons un bilan avec une étude comparative pour différentes initialisations de paramètres en le confrontant aux performances d'algorithme de type « plus court chemin ».

Nous terminerons notre travail d'investigation par une conclusion générale et des perspectives. Les algorithmes d'apprentissage par renforcement issus de la méthode différence temporelle (TD) sont détaillés en annexe. Une présentation concise du logiciel Net Logo (environnement de simulation basé agent) utilisé lors de la validation de notre approche de conception est également jointe.

Partie A

De l'Agent …

au Système Intelligent

Partie A

De l'agent … au système Intelligent

Introduction

Nous avons introduit l'Intelligence Artificielle comme ayant trait à la conception d'entités intelligentes. Nous avons cité aussi la rationalité comme concept de base. Nous pouvons alors envisager l'Intelligence Artificielle comme la conception d'agents rationnels. Avant d'aller plus loin dans la présentation de notre travail, il nous semble nécessaire de définir plus précisément un certain nombre de notions telles que la notion d'agent et de système intelligent.

L'objet de cette première partie est donc d'introduire ces concepts qui seront à la base du travail que nous présenterons par la suite. Nous commencerons, dans un premier chapitre, par définir la notion d'agent et les différents concepts indissociables puis, nous aborderons en bref la dimension sociale de ces entités et terminerons par décrire quelques travaux qui ont été effectués dans le domaine. Nous exposerons, dans le deuxième chapitre, les problèmes liés à la décision dans les systèmes intelligents et soulignerons l'importance du critère d'apprentissage dans la mise en place de ces entités dites « rationnelles ».

Chapitre I

Agent …

Chapitre I

Agent

1.1 Introduction

Bien que l'Intelligence Artificielle (IA) ait fait l'objet de nombreuses définitions et puisse être abordée selon différents points de vue, il est possible de s'accorder sur le fait qu'elle consiste en la mise en place de systèmes intelligents également appelés agents.

La notion d'agent occupe donc une place prépondérante en IA et va constituer la base de ce premier chapitre. Nous commencerons par présenter la diversité des définitions de ce terme et développerons les concepts d'environnement et d'autonomie qui y sont étroitement liés ainsi que le mode de fonctionnement et les principales applications des systèmes à base d'agents

1.2 Présentation du concept « Agent »

La notion d'« agent » est relativement récente et plusieurs domaines de recherche ont contribué à son apparition. Les principaux travaux, qui ont été à l'origine de ce concept, appartiennent aux domaines suivants [Jenning et *al*.,1998 ; Chaib-Draa, 1999] :

- ✓ L'intelligence artificielle (IA) [Russell et Norvig, 1995].
- ✓ La programmation orientée objet et systèmes à base de données concurrents [Agha et *al.*, 1993].
- ✓ Les langages d'acteurs [Hewitt et *al.*, 1973 ; Agha, 1986 ; Agha et *al*.,1988].
- ✓ La conception d'interface homme/machine [Maes, 1994].

Mais c'est plutôt le domaine de l'IA qui a participé le plus à l'évolution du paradigme agent. En effet la plupart des définitions ont été établies par des chercheurs en IA.

A l'heure actuelle, aucune définition n'a été universellement adoptée bien que certaines réunissent la majorité des opinions. L'étude de ces définitions permet toutefois d'en identifier des traits communs. Franklin et Graesser donnent un grand aperçu dans [Franklin et Graesser, 1996] et proposent une classification des variétés d'agents existants selon leurs caractéristiques.

1.2.1 La multiplicité des définitions

Pour avoir une bonne vision, nous allons donner quelques définitions de l'aspect mono-agent ou formellement « agent situé » proposées dans la littérature.
Intéressons nous tout d'abord à une définition très générale du mot dans le dictionnaire de la langue française.

Définition 1 : selon le dictionnaire « Larousse » : Ce mot qui vient du latin « agere» signifie « faire, mettre en mouvement » ou l'être qui agit. Ce qui agit, opère ; force, corps, substance intervenant dans la production de certains phénomènes.

Comme l'évoque cette définition, l'action est au cœur du concept agent. Cette action est également à la base de la définition de Russel et Norvig présentée dans leur ouvrage [Russel et Norvig, 2003] introduisant les principes fondamentaux de l'intelligence artificielle et qui constitue une bonne entrée en la matière.

Définition 2: Selon Russel et Norvig: « **An agent is just something that acts. »** ***[Russell et Norvig 2003]: Un agent est simplement quelque chose qui agit.***

La première question qui se pose lorsque nous évoquons cette définition consiste à établir une différenciation entre un agent et un programme informatique. Selon Russell et Norvig, un agent est quelque chose qui agit, nous pouvons alors supposer qu'un programme informatique est un agent : Il reçoit des informations en entrée et produit un résultat qui est retourné à la fin de son exécution. Un programme agit et pourrait donc être assimilé à un agent. Il existe cependant d'autres propriétés fondamentales qui distinguent un simple programme d'un agent et que Russell et Norvig ajoutent à leur définition: « *But computer agents are expected to have other attributes that distinguish them from mere programs, such as operating under autonomous control, perceiving environment, persisting over a prolonged period, adapting to change, and being capable of taking on another's goals »*

C'est ainsi qu'un agent informatique est supposé posséder ces caractéristiques qui les distinguent de simples programmes, comme l'autonomie, la perception de l'environnement, la persistance sur une période prolongée, l'adaptation au changement et la faculté de modifier

ses buts.

Cette définition complète et enrichit celle donnée par les même auteurs dans [Russell et Norvig 1995] : "*Un agent est tout ce qui peut être compris comme percevant son environnement à travers des senseurs et comme agissant sur cet environnement par l'intermédiaire d'effecteurs"*.

Dans cet ouvrage **[**Russel et Norvig, 1995**]**, ils définissent aussi la notion d'autonomie par rapport à la quantité d'informations dont dispose initialement l'agent. Si le comportement d'un agent est déterminé à priori, Russel et Norvig affirment que l'agent manque alors d'autonomie.

Les notions d'autonomie, d'environnement et d'objectif, sont également présentes dans la définition donnée par Wooldridge et Jennings [Wooldridge et Jenning, 1995] :

Définition 3 « An agent is a computer system that is **situated** in some environnment, that is capable of **autonomous** and **flexible** action in this environment in order to meet its design objectives » : ils définissent « un agent comme étant un système informatique situé dans un environnement et dans lequel il est capable d'agir de façon souple et autonome afin de satisfaire les objectifs pour lesquels il a été conçu ».

Revenons en détail sur ces propriétés [Weiss, 1999] :

Un agent est dit « **situé »**, s'il perçoit son environnement via ses capteurs, et peut en changer la configuration en agissant dessus via ses effecteurs. Le terme **situé** indique la capacité d'interagir avec son environnement.

Par opposition donc à l'intelligence artificielle symbolique traditionnelle qui conçoit des programmes informatiques déconnectés de leurs environnements, l'intelligence artificielle située (IAS) se fonde sur le postulat qu'un système intelligent ne doit pas baser son comportement sur les raisonnements strictement abstraits effectués dans un système clos, mais doit agir et se comporter de façon à assurer sa survie dans son monde environnant [Sigaud, 1999 ; Drogoul, 1999 ; Baquiast, 2001]**.**

Ainsi le comportement de l'agent est en adaptation continue avec la représentation qu'il se fait de l'environnement considéré comme complexe et il est autonome dans le sens où sa conduite est déterminée par ses propres expériences. Cette définition nous permet ainsi d'introduire la notion d'interaction entre l'agent et l'environnement :

L'interaction peut être définie aussi comme étant l'influence mutuelle entre ces deux entités (agent et environnement) et qui est établie via les processus de perception de l'agent de son environnement et de son action sur ce dernier.

Elle peut être directe, comme dans le cas de l'action de l'agent sur l'environnement; c'est une modification explicite de l'état de ce dernier. Elle peut aussi être indirecte, comme dans le cas de la perception de l'environnement par l'agent; l'environnement ne peut pas modifier lui-même l'état de l'agent, mais ce dernier peut changer d'état en fonction des percepts récupérés à partir de l'environnement [Chaib Draa, 2002].

Cette notion est donc fondamentale pour implanter celle de la réactivité. Celle-ci, à son tour, garantit que l'agent réplique dans certaines situations imprévues rapidement et correctement pour assurer son autonomie. Et enfin, sans cette dernière, l'agent se réduit à un simple « objet » informatique qui possède des données et offre des méthodes [Florea et *al*., 2002 ; Odell, 2002; Luck et *al*., 2004].

Concernant la propriété de flexibilité énoncée dans la définition, et qui dérive de celle de l'autonomie, elle sera acquise lorsque l'agent vérifie les caractéristiques suivantes :

- ✓ La réactivité, qui est la capacité de répondre aux événements extérieurs,
- ✓ La pro-activité, qui est la capacité de prédire et d'anticiper les changements dans l'environnement et d'agir en fonction d'eux.
- ✓ La sociabilité, qui est la capacité de communiquer et d'interagir avec les autres.

Une autre définition donnée par Jean Sallantin dans son ouvrage, « Agent Intelligent » [Sallantin, 1997].et qui associe un agent à :

Définition 4 : *« un système modifié par son interaction avec son environnement* ».

Ainsi, un système qui a la capacité de s'auto-modifier en'' apprenant '' pour atteindre ses objectifs est appelé « agent ».

Ferber propose également dans [Ferber, 1995] une définition de la notion d'agent dans laquelle nous retrouvons ces concepts clefs. Sa définition est cependant plus riche.

Définition 5 : Ferber considère *l'agent comme étant une entité physique ou virtuelle:*

1 - qui est capable d'agir au sein d'un environnement E,

2 - qui peut communiquer directement avec d'autres agents,

3 - qui est mue par un ensemble de tendances, sous la forme d'objectifs individuels, d'une fonction de satisfaction ou de survie, qu'elle cherche à optimiser,

4 - qui possède des ressources propres,

5 - qui est capable de percevoir, mais de manière limitée, son environnement,

6 - qui ne dispose que d'une représentation partielle de cet environnement et

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 dispose cette entité et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.

Tandis que les items 1 et 5 reprennent les principes d'action et de perception comme nous l'avons vu concernant la définition d'un agent, Ferber intègre d'autres caractéristiques intéressantes : Il distingue principalement la capacité de communiquer avec les autres agents (2), la prise de décisions en accord avec ses objectifs (3) en tenant compte de ses ressources propres (4) et enfin la propriété de localité de l'agent (6) influe sur ses connaissances de l'environnement. Les décisions de l'agent sont dirigées par ses tendances, en accord avec les ressources disponibles et les connaissances acquises.
Dans cette définition, l'agent est autonome et son évolution est liée à celle de l'environnement.

1.2.2 Les concepts communs

A partir des définitions présentées, nous pouvons mettre en évidence le fait que la notion d'agent se base sur trois concepts communs : l'agent lui même, l'environnement et l'autonomie. Nous allons à présent revenir plus en détail sur chacun de ces trois concepts.

- ✓ **Agent**

Toutes les définitions du concept d'agent informatique, s'accordent sur le fait qu'un agent est une entité équipée de capteurs lui permettant de percevoir son environnement et d'effecteurs lui permettant d'agir dans cet environnement.

A partir de ces capteurs, l'agent acquiert des connaissances sur son environnement qui peuvent enrichir celles fournies à priori par le concepteur. Ces informations lui permettent alors de décider comment agir. Cette décision découle d'un processus plus au moins complexe : un agent peut, par exemple, baser ses décisions sur ses perceptions courantes, ou bien sur l'historique de ses perceptions. Après délibération, l'agent exécutera, grâce à ses effecteurs, l'action qu'il a décidé de réaliser.

- ✓ **Environnement**

L'environnement joue un rôle particulièrement important dans cette définition. En effet, ce dernier peut être vu comme un ensemble de ressources externes dont dispose l'agent

et de règles, qui guident son évolution et qui subissent les conséquences de ses actions [Bouzid, 2001]. Les données contenues dans l'agent représentent les objectifs visés, et celles intégrées dans l'environnement sont les ressources externes dont dispose l'agent. La notion clé de cette structuration est l'interaction entre l'agent situé et l'environnement, qui n'est autre que la perception et l'action locales de l'agent sur son environnement.

Ce concept est essentiel dans notre problématique de conception d'agent intelligent. L'environnement produit les conditions nécessaires à l'existence de l'agent et définit les propriétés du monde dans lequel il fonctionne [Odell et *al* , 2002].

- ✓ **Autonomie**

Considérons un agent placé dans un environnement et devant décider, à partir de ses perceptions, quelle action exécuter. Si cette décision est prise sans l'intervention d'un tiers, l'agent est dit autonome.

Cette définition de l'autonomie caractérise un agent qui base ses décisions uniquement sur ses connaissances acquises, donc capable de s'affranchir de celle de son concepteur. Mais, l'agent doit être en mesure d'acquérir et d'engranger des informations et des données afin de pallier à son manque de savoir initial, ce qui lui permettra d'améliorer son comportement en tenant compte d'expériences passées [Bouzid et *al*., 2001].

1.3 Architecture interne d'un agent

Un autre aspect est jugé important lors de la définition d'un agent. C'est la description de son architecture. Par opposition à l'architecture externe, regroupant les capteurs et les effecteurs dont l'agent est doté, l'architecture interne d'un agent désigne l'ensemble des structures de données et des processus internes lui permettant de prendre une décision. Elle se compose des mécanismes qui régissent son comportement. A ce titre, une distinction classique se dégage dans le domaine entre les agents réactifs et les agents cognitifs [Ferber, 1997 ; Drogoul, 1999].

Les agents réactifs, ne disposant pas de représentation interne et explicite de l'environnement, réagissent de façon réflexe et appuient leur comportement sur un ensemble de stimuli-réponses. Chaque stimuli (ou situation) fait correspondre une réponse (ou action). Ce processus de délibération étant simple à appliquer, ces agents font preuve d'une grande réactivité, d'où leur nom d'agents réactifs ou d'agents « réflexes ».

Les agents cognitifs disposent de capacités de raisonnement développées. Ils sont caractérisés par : une représentation explicite de leurs objectifs, une représentation évoluée de l'environnement, et une capacité à manipuler ces représentations pour anticiper ou réévaluer

ces objectifs. Ils sont capables donc de planifier sur la base d'un modèle de leur univers.

Certains chercheurs affirment qu'il est difficile de travailler avec une architecture typiquement réactive ou cognitive et suggèrent, au contraire, d'utiliser des agents hybrides [Ferguson, 1992 ; Müller, 1997] ayant des capacités délibératives et réactives. Ils conjuguent en effet la rapidité de réponse des agents réactifs ainsi que les capacités de raisonnement des agents cognitifs.

Les architectures BDI (Belief Desire Intention) constituent un type d'architecture d'agents cognitifs [Bratman et *al*, 1988]. Cette architecture est basée sur les notions d'attitudes mentales que sont la Croyance (Belief), le Désir (Desire) et l'Intention (Intention) : les croyances correspondent aux informations (éventuellement incomplètes et incorrectes) qu'a l'agent de son environnement, les désirs correspondent aux états de l'environnement que l'agent souhaiterait voir réalisés et les intentions correspondent aux projets de l'agent pour satisfaire ses désirs. Les prises de décisions d'un agent BDI sont alors effectuées à partir de la manipulation des états mentaux de l'agent et de la révision de ceux-ci au cours du temps [Shoham 1993 ; Rao et George, 1995].

Discussion

Les agents réactifs sont les plus simples, et leur efficacité a été estimée dans plusieurs applications [Brooks, 1986 ; Dutech, 1999 ; Buffet, 2000, Drogoul, 2001, Buffet, 2003]. Mais bien que leur comportement soit « élémentaire », ils ne sont pas évidents à implanter. Au contraire, les agents cognitifs sont beaucoup plus compliqués, plus difficiles à mettre en œuvre.

Les agents hybrides ont essayé de combiner le meilleur des avantages des deux architectures précédentes et les agents BDI essayent de reproduire le raisonnement humain mais ne sont pas convenables pour les mêmes raisons que celles des agents cognitifs.

Brooks dans [Brooks, 1991] s'oppose à l'utilisation d'agents cognitifs pour construire des systèmes intelligents. Il affirme que "la complexité du comportement d'un système ne réside pas dans la complexité de l'agent mais dans celle de l'environnement et de l'interaction qu'entretien l'agent avec ce dernier".

Selon lui, concevoir des agents intelligents cognitifs se heurte au problème de la représentation explicite de l'environnement au sein de l'agent. Il propose plutôt d'utiliser directement les perceptions de l'agent pour la prise de décision et de construire des agents intelligents de manière incrémentale en commençant par des agents aux comportements très élémentaires.

1.4 Mémoire de l'agent

Pour pouvoir garder une trace de ses expériences et adapter son comportement en conséquence, un agent a besoin de mémoire.

Deux principaux types de mémoire peuvent être rencontrés :

- la mémoire à court terme qui est destinée à stocker des événements précis afin de modifier les décisions de l'agent en fonction d'observations passées. Ce type de mémoire intervient directement comme entrée des règles stimuli- réponse de l'agent.

- 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 internes à l'agent et d'en produire éventuellement de nouvelles. Le problème posé 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.

Discussion

Pour des agents qui sont autonomes, l'application plutôt d'une forme de mémorisation est à préconiser puisque l'agent intégrera automatiquement chaque nouveau terme appris dans son processus de décision. Une question reste cependant posée : celle consistant à déterminer l'élément utile à mémoriser.

1.5 Fonctionnement d'un agent

De manière plus précise, le fonctionnement général d'un agent est décrit par une boucle fermée appelée boucle sensori-motrice (ou perception - action) :

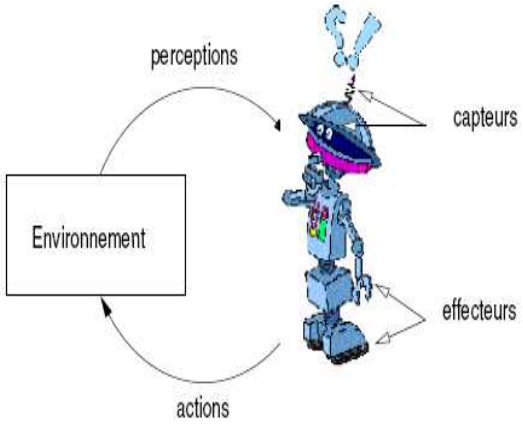


Figure 1.1 - Définition d'un agent au sens de Russell et Norvig

[Russel et Norvig, 1995]

L'exécution de cette boucle peut se décomposer en plusieurs étapes :

- ✓ Initialement, l'agent se trouve dans une certaine configuration interne.
- ✓ Il perçoit une partie de l'environnement dans lequel il est plongé grâce à ses capteurs.
- ✓ Il choisit une action à entreprendre en fonction de sa configuration interne et de ses perceptions. Ce choix sera considéré comme le résultat d'une fonction de prise de décision
- ✓ Il effectue cette action dans le but de modifier l'état de l'environnement ou/et sa configuration interne.
- ✓ Il reçoit de nouvelles perceptions et le processus se répète.

A partir de là, on peut distinguer l'importance de la mémorisation pour introduire la faculté de raisonnement, de décision et d'apprentissage au niveau de l'agent

1.6 Notion d'un système multi-agents (SMA)

Un système multi-agents peut être défini comme étant un ensemble organisé d'agents. Un agent peut constituer à lui seul un système multi-agents selon le principe de récursivité, et un système multi-agents peut être considéré comme un seul agent à un niveau supérieur d'abstraction.

1.6.1 Définition

Un SMA est un système dans lesquels des agents artificiels opèrent collectivement et de façon décentralisée pour accomplir une tâche. Ceci signifie que dans un système multi-agents, il existe une ou plusieurs organisations qui structurent les règles de cohabitation et de travail collectif entre les agents (définition des différents rôles, partage de ressources,...). Les agents sont généralement situés dans un environnement contenant également des entités passives communément appelées objets.

1.6.2 Organisation

En 1995, une approche intégrée (l'approche « Vowels ») des systèmes multi-agents est proposée par Yves Demazeau [Demazeau, 1995]. Elle est basée sur une décomposition en quatre parties:

- ✓ **Agents A :** qui concernent les modèles ou architectures utilisés pour la partie active de l'agent, depuis un simple automate, jusqu à un système complexe à base de connaissances ;

✓ **Environnement** E : représente les milieux dans lesquels évoluent les agents. Ils sont généralement spatiaux ;

✓ **Interactions I :** qui englobent les infrastructures, les langages et les protocoles d'interactions entre agents ;

✓ **Organisation O :** Concerne la structuration des agents en groupes : hiérarchies, relations...

Demazeau [Demazeau, 1995] définit le système multi-agents selon ces cinq aspects. L'interaction est ainsi un des aspects clés des systèmes multi-agents. Elle offre un moyen pour assurer la coopération et la négociation entre agents. Sans interaction (ou communication), l'agent n'est qu'un individu isolé, renfermé sur sa boucle perception – délibération - action [Ferber, 1995]. Bien que l'interaction et la communication sont souvent confondues dans la littérature, la communication représente la transmission d'informations entre agents, alors que l'interaction comprend l'action sur le monde, ainsi que la communication entre les agents du système [Briot, 2001].

De plus, cette approche est guidée par deux principes, l'approche déclarative et fonctionnelle :

✓ **L'approche déclarative** qui peut se résumer à travers cette équation :

S M A = Agents + Environnement + Interactions + Organisation

✓ **L'approche fonctionnelle** : Qui permet de donner les fonctionnalités des SMA : Ces dernières incluent les fonctionnalités individuelles des agents enrichies des fonctionnalités qui résultent de la valeur ajoutée par le système multi-agents lui-même, parfois appelée intelligence collective. Une deuxième équation peut alors être formulée :

Fonction (SMA) =Σ Fonction (AGENT) + FonctionCollective

Ferber donne une vision plus large au système multi-agents, qui doit être composé des éléments suivants [Ferber, 1995]

✓ Un environnement E, c'est-à-dire un espace ;

✓ Un ensemble d'objets O : Ces objets sont situé, c'est-à-dire que pour tout objet passif, il est possible d'associer une position dans E;

✓ Un ensemble d'agent A : Représentent les entités actives du système ;

✓ Un ensemble d'opération Op permettant aux agents de percevoir, produire et consommer et manipuler les objets.

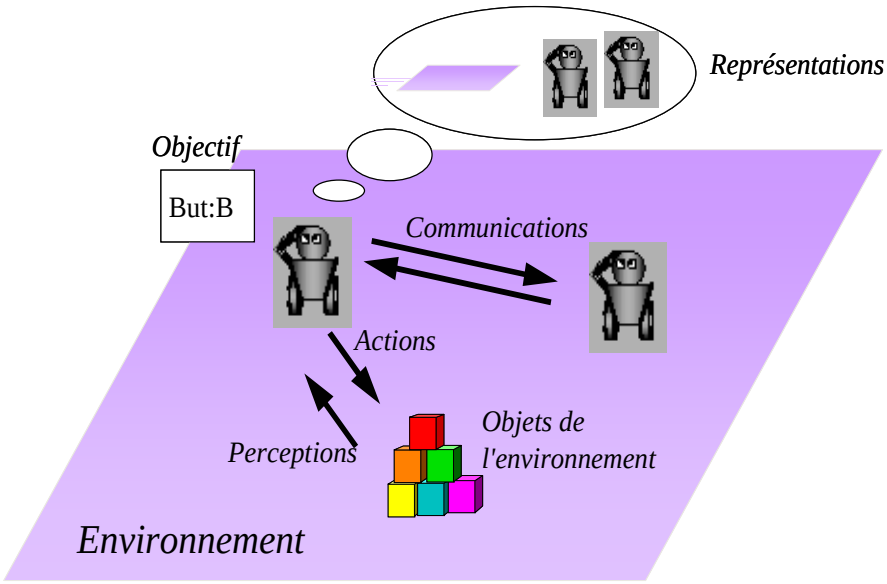


Figure 1. 2 - Système multi-agents selon [Ferber, 1995]

Jenning donne aussi un sens plus étendu à ce terme (SMA), qui permet de l'utiliser pour tous types de systèmes formés de plusieurs composants autonomes respectant les caractéristiques suivantes [Jenning et *al*, 1998] :

- ✓ Chaque agent a des possibilités limitées pour résoudre un problème;
- ✓ Il n'existe pas de contrôle global du système ;
- ✓ Les données sont décentralisées ;
- ✓ Le calcul est asynchrone.

Discussion

Enfin, conformément au point de vue généralement utilisé, un système multi-agents (SMA) est constitué d'un ensemble de processus informatiques se déroulant en même temps, donc de plusieurs agents vivant au même moment, partageant des ressources communes et communicant entre eux. Le point clé des systèmes multi-agents réside dans la coordination entre les agents. La recherche sur les agents est ainsi une recherche sur :

1. **la décision :** comment décomposer l'objectif global en sous buts et tâches ?
2. **le contrôle**: quelles sont les relations entre les agents et comment sont-ils coordonnés?
3. **la communication:** différents protocoles sont proposés en fonction du type de coordination entre les agents.

Les systèmes multi-agents ont des applications dans le domaine de l'intelligence artificielle où ils permettent de réduire la complexité de la résolution d'un problème en divisant le savoir nécessaire en sous-ensembles, en associant un agent intelligent à chacun de ces sous-ensembles et en coordonnant l'activité de ces agents [Ferber, 1995]. On parle ainsi d'intelligence artificielle distribuée.

Dans le cadre de notre travail, on ne s'intéresse pas à la dimension sociale de l'agent mais à l'agent lui-même.

1.7 Agent : principaux travaux

Avant de conclure cette partie, nous tenons à réaliser une ébauche de la diversité des applications des systèmes à base d'agent. Au début de l'apparition des agents, les chercheurs se sont concentrés principalement sur la conception de modèles de raisonnement, afin de les doter de capacité cognitive. Ces modèles permettent de définir comment l'agent décide de ses actions en fonction de ses percepts et de ses objectifs. L'intérêt pour ces systèmes s'est accru de façon importante ces dernières années. Woooldridge [Wooldridge, 2002] détaille un certain nombre d'applications et propose une méthodologie pour la mise en oeuvre de ces systèmes: pilotage automatique d'avions, recherche d'informations, e-learning et autres.. Les environnements virtuels peuvent aussi utiliser l'approche agent afin d'étudier le comportement d'individus. Les applications liées à l'Internet font également appel à de tels systèmes : recherche d'information, indexation de données, aide à la navigation, enchères et commerce électroniques, …

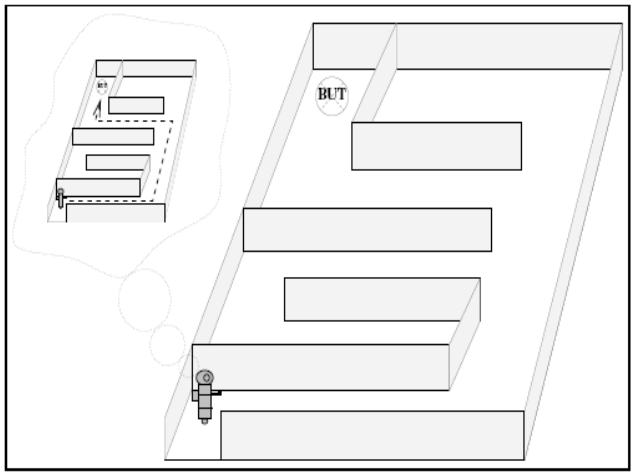


Figure 1. 3- Agent robot

1.8 Conclusion

La notion d'agent situé est une nouvelle approche pour aborder la modélisation et la résolution de problèmes dans le domaine de l'intelligence artificielle. Son avantage est qu'elle offre un bon cadre pour la structuration des informations, représentées par l'environnement et l'agent, et d'intégrer le raisonnement au sein de ce dernier.

Nous venons d'étudier quelques unes des nombreuses définitions de la notion d'agent. Nous avons également présenté les principaux concepts indissociables de cette notion, et avons pu remarquer que la définition d'une même notion pouvait différer d'un auteur à un autre. Et malgré la diversité des définitions recensées, nous avons réussi à identifier les caractéristiques importantes d'un agent.

Quelle que soit l'interprétation du mot « agent » considérée, nous pouvons remarquer qu' « Agent et environnement » sont étroitement liés : tout agent est situé dans un environnement qu'il perçoit et dans lequel il agit.

L'autonomie quant à elle est une caractéristique majeure des agents qui permet de les différencier des simples programmes informatiques et du concept d'objet. Un agent n'a pas accès à l'information utile et ne connaît pas forcément la configuration globale du système, en contrepartie, il doit contrôler ses propres actions ainsi que son état interne. Nous nous concentrerons sur des agents autonomes selon cette définition.

Il reste à définir la notion d' « intelligence » et les critères d'agent « intelligent » qui vont être présentés dans le chapitre suivant.

Chapitre II

…Système Intelligent

Chapitre II

Système Intelligent

2.1 Introduction

L'intelligence est une notion difficile à cerner, même appliquée à un individu, elle est mal définie. Derrière ce terme se cachent de nombreuses acceptions différentes. Si nous sommes tous d'accord pour dire que l'homme, d'une manière générale, est intelligent, il nous est plus difficile de déterminer exactement les limites de cette intelligence même si nous sommes capables de juger si un acte, une idée ou encore une solution à un problème donné le sont.

En effet, la tendance générale est d'associer le degré d'intelligence à la difficulté de résolution d'un problème et l'estimation de cette difficulté est dépendante de l'observateur, de l'environnement et de la tache. Un système pourra être qualifié d'intelligent parce que, pour un observateur extérieur, il semblera doté de capacités cognitives habituellement attribuées à l'homme (intelligence émulée) ou parce qu'il cherchera à reproduire les mécanismes par lesquels l'homme ou l'animal prend des décisions complexes (intelligence simulée) [Thomas, 2005]

2.2 Concept d'Agent Intelligent

Concernant un système intelligent, on retrouve dans [Piaget, 1967], la définition suivante : « Capacité à s'adapter effectivement à l'environnement soit en changeant soi-même, soit en changeant l'environnement, soit en trouvant un nouveau » C'est donc un caractère important puisqu'il représente la capacité d'adopter un comportement souple et adaptatif dans un environnement imprévisible et inconnu.

Une autre définition stipule que **: «** Pour pouvoir être qualifié d'intelligent, un système artificiel doit être doté de capacités d'apprentissage » [Goldman, 1996 ; Sallantin,

1997; Vercouter et *al.*, 1998]. C'est également cet aspect très important d'apprentissage qui permet d'acquérir de nouvelles connaissances et de les faire évoluer et qui est nécessaire au développement de ces systèmes.

L'apprentissage situé est une approche qui considère que l'agent doit effectuer sa « formation » dans le monde où il fonctionne en partant des contraintes de l'environnement dans lequel la tache est à réaliser [Pfeifer et *al.*, 1999] .

De plus, lorsqu'un système est construit, il est difficile et même impossible de déterminer, à priori, correctement un comportement «optimal» qui pourrait répondre à toutes les problématiques et situations qu'il pourrait rencontrer. Si nous souhaitons alors en développer un qui puisse s'affranchir de l'intervention d'un tiers pour être qualifié d'intelligent, il serait nécessaire de mettre en place ces mécanismes décisionnels flexibles «nommés apprentissage» qui lui permettent de faire face, tout seul, à un grand nombre de situations.

2.3 Notion d'apprentissage pour un système artificiel

Le domaine de l'apprentissage a émergé depuis plusieurs années au sein de la communauté d'agents, suite à un réel besoin de techniques et de méthodes visant à améliorer les performances de ces systèmes. Du fait de leurs complexités de conception, l'apprentissage, apparaît donc comme une solution efficace. Chaque agent peut ainsi apprendre entre autres, les comportements stratégiques et réagir plus facilement à chacune des situations afin de maximiser ses performances.

2.3.1 Définition de l'apprentissage

D'une manière très générale, on définit l'apprentissage comme étant l'ensemble de règles de modification du système en fonction des données observées [Munos, 2007]. C'est aussi « la capacité à faire mieux la prochaine fois » [Dutech, 1999]. Les termes utilisés dans cette définition sont assez flous et exigent une signification plus explicite.

- ✓ **Faire mieux** : Soit que le système devient plus efficace, plus rapide, soit qu'il capitalise sur l'augmentation de ses connaissances et de ses capacités à raisonner. Ainsi, un système apprend s'il améliore sa capacité à résoudre des problèmes en utilisant de l'information obtenue par interaction sur son environnement ou bien s'il construit ou modifie une représentation interne de ses expériences.
- ✓ **La prochaine fois** : la majeure partie des travaux de recherche utilise des

expériences qui sont exactement reproductibles dans le temps. La prochaine fois laisse sous entendre que le système devra savoir reconnaître des problèmes et des situations semblables pour pouvoir les intégrer à sa représentation du monde sans avoir à traiter chaque cas comme un cas particulier.

2.3.2 Différents types d'apprentissage

L'apprentissage peut être centré selon quatre axes : agent, environnement, interaction ou organisation [Vercouter et *al.*, 1998].

1- Apprentissage centré agent : Concerne ce qu'un agent peut apprendre sur lui-même ou sur les autres agents. Cet apprentissage porte sur le comportement de l'agent, ses stratégies et ses décisions

2- Apprentissage centré environnement : se focalise sur ce que l'environnement peut apprendre sur l'agent. Cela peut porter sur nombreux objets tant la diversité des environnement peut être grande. Dans un environnement à forte dynamique, un agent peut apprendre sur les parties nouvellement apparues (vie artificielle)

3- Apprentissage centré interaction : porte sur les moyens mis en œuvre par les agents pour communiquer ou interagir (nouveaux langages, communications.)

4- Apprentissage centré organisation : s'occupe de faire évoluer les rôles des agents au sein de leur société. Lorsque le système est supervisé, les agents sont guidés dans leur démarche, les agents autonomes quant à eux doivent apprendre à s'auto-organiser (vie artificielle).

2.3.3 Méthodologie d'Apprentissage

Il nous paraît aussi important de différencier les méthodes d'apprentissage selon le rôle du "professeur", c'est-à-dire la façon dont les données d'apprentissage sont fournies à l'apprenant [Dutech, 1999] :

1- Apprentissage supervisé (avec professeur) : On dispose d'une base d'apprentissage associant à un ensemble de situations la réponse correcte. Le but est d'induire une fonction permettant de généraliser à des situations non présentes dans l'ensemble d'apprentissage. La méthode supervisée doit pouvoir disposer d'exemples étiquetés, donc ne peut s'appliquer qu'à des problèmes qui ont déjà été résolus. Le rôle de l'apprentissage est alors de déterminer la règle suivie par le professeur pour associer les labels aux données.

2- Apprentissage non supervisé (sans professeur): on dispose simplement d'un ensemble de données, le but est alors de trouver des similarités entre ces données en

identifiant des regroupements ou des prototypes". Les méthodes non supervisées utilisent des exemples bruts et essaient d'y détecter des régularités.

3- Apprentissage par renforcement: c'est une méthode permettant d'apprendre à résoudre une tâche donnée simplement en donnant un signal sous forme de récompense ou de punition à l'agent. On n'a pas à dire explicitement comment résoudre cette tâche mais simplement si ce que fait l'agent est plus ou moins bien. Les méthodes avec renforcement demandent au professeur d'avoir un rôle passif: il ne fait que juger de la qualité de réponse de l'apprenant. Souvent, c'est l'environnement lui même qui peut fournir cette information à l'apprenant.

Nous résumons les caractéristiques des trois types d'apprentissage dans le tableau suivant [Munos, 2007]

	Supervisé	Non-supervisé	Par renforcement
Qu'est ce qu'on apprend ?	**Relations**	**Structures**	**Loi d'action**
Info pour l'apprentissage	**Sortie désirée**	**Rien**	**Renforcement**
Forme d'apprentissage	**Par Instruction**	**Par observation**	**Par évaluation**
Loi d'apprentissage	**Gradient**	**Auto-organisation**	**Différence**
But de l'agent	**Faire la bonne action**	**Structurer ses perceptions**	**Maximiser ses récompenses**
Perception	**L'action qu'il fallait faire**	**Aucun indice**	**Récompenses ou punitions**

Tableau 2.1- **Tableau Récapitulatif des méthodologies d'apprentissage**

Discussion :

La notion d'apprentissage n'est pas complètement définie mais reste tout de même moins difficile à appréhender que la notion d'intelligence. Etant liée à la construction et à la modification de représentations du monde, elle permet d'accroître les capacités d'adaptation d'un système, le rendant plus autonome et plus rationnel [Dutech, 1999; Thomas, 2005].

Il est à noter qu'il existe une différence entre apprentissage et adaptation. En effet, nous considérons, qu'il y a apprentissage lorsqu'un système acquiert de nouvelles connaissances et devient plus performant alors qu'il y a adaptation lorsqu'un système fait face à de nouvelles conditions sans apprendre de nouvelles choses et sans pour autant devenir plus efficace. Bien que la distinction paraisse assez floue, il est à signaler qu'un système ne pourra s'adapter s'il n'apprend pas sur ces expériences vécues [Mitchell, 1997].

2.4 Propriétés attendues d'un agent intelligent

Russel et Norvig [Russel et Norvig, 2003] fournissent aussi une taxonomie des systèmes "intelligents" selon deux axes de classification :

- Le premier axe traite de l'objet de l'intelligence : qui distingue les systèmes pour lesquels l'intelligence réside dans les raisonnements et les manipulations de représentations internes et les systèmes pour lesquels l'intelligence réside dans le comportement du système et les interactions que ce dernier entretient avec son environnement.

- Le second axe caractérise la façon dont la notion d'intelligence sera évaluée : l'intelligence d'un système pourra être évaluée selon la conformité du comportement/raisonnement artificiel par rapport à des comportements/raisonnements humains ou de manière objective par rapport à des critères numériques de performance : on parlera alors de système rationnel.

En suivant cet objectif, on cherche à construire des systèmes qualifiés d'intelligents du fait des actions qu'ils émettent et des interactions qu'ils entretiennent avec l'environnement dans lequel ils sont plongés, et ce par rapport à un critère qualitatif le plus souvent numérique. C'est ce qu'on appelle « système rationnel »

Discussion

C'est ainsi que la problématique entourant la notion de système intelligent implique donc la conception d'un système rationnel ayant des capacités particulières de perception, de traitement et de prise de décision et d'action lui permettant de s'adapter avec une certaine autonomie à son milieu d'opération afin d'y remplir efficacement son rôle.

2.4.1 La Rationalité

Newell est le premier a avoir parlé d'agent rationnel pour évoquer un système [Newell, 1982]. Selon lui, l'agent est doté de fins (l'ensemble des buts), de moyens physiques d'interaction avec le monde extérieur (l'ensemble des interactions, compétences et de connaissances), liant buts et moyens selon le principe de rationalité.

La rationalité constitue donc la manière dont la notion d'intelligence peut s'exprimer au niveau de l'agent [Weiss, 1999].

Une première caractérisation relativement naïve stipule qu'un agent rationnel est un agent qui "effectue les bonnes actions au bon moment" [Russel et Norvig, 1995]. Afin de

raffiner cette caractérisation, Russel et Norvig préconisent dans [Russel et Norvig, 2003] d'utiliser une mesure de performance pour caractériser l'objectif de l'agent et évaluer les actions qu'il a pu choisir. Cette mesure de performance constitue une représentation du problème à résoudre. Elle est indépendante de l'agent et nécessite d'être définie par un observateur extérieur au système qui peut analyser dans quelle mesure le système parvient à répondre au problème posé. Une fois une mesure de performance établie, c'est à l'agent rationnel d'exprimer un comportement apte à maximiser cette mesure dans le long terme. [Thomas, 2005].

La mesure de performance tient compte des connaissances de l'agent, de ses perceptions et des actions exécutées. Russell et Norvig définissent alors un agent rationnel de la façon suivante :*« For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has» [Russel 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 perceptions et quelles que soient ses connaissances à priori ».

Qualifier une décision de rationnelle sera donc fonction de :

- ✓ La mesure de performance définissant l'objectif de l'agent ;
- ✓ L'historique des perceptions de l'agent ;
- ✓ Les connaissances qu'a l'agent de l'environnement ;
- ✓ Les actions que l'agent peut effectuer.

De cette rationalité se dégagent aussi les notions qui vont suivre et qui influencent essentiellement le comportement de l'agent. Elles conditionnent son « degré d'intelligence » et auront donc un impact sur la manière de le concevoir:

2.4.2 La Réactivité

C'est la capacité de l'agent à percevoir son environnement et à décider comment agir en respectant le temps qui lui est accordé et dont le processus de délibération est basé sur des règles stimuli $\rightarrow$ réponse. De manière générale, plus un mécanisme délibératif sera simple, plus l'agent fera preuve d'une grande réactivité. Un tel agent est en mesure de prendre ses décisions en temps réel.

2.4.3 L'Adaptation

Elle englobe l'ensemble des activités par lesquelles un agent modifie ses conduites pour s'ajuster de manière optimale à un milieu déterminé [Piaget, 1967]. Ce processus d'adaptation peut prendre diverses formes :

- ✓ Il peut s'agir d'un raisonnement construit sur un ensemble de connaissances à priori et en produisant d'autres ;
- ✓ Ou bien l'acquisition de connaissance qui va permettre d'atteindre le but fixé en passant par un apprentissage par l'expérience.

Finalement, si le système peut continuer à effectuer sa tâche dans un environnement inconnu à priori, on dit qu'il est capable de s'adapter.

2.4.4 La pro-activité

Elle caractérise la capacité de l'agent de se fixer des buts afin d'atteindre des objectifs. Son comportement n'est ni une simple réponse aux changements de l'environnement ni les effets immédiats de ses actions. Un agent pro-actif peut planifier ses actions afin de satisfaire ses buts.

2.4.5 Le Degré de délibération

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.

Suivant les spécificités des applications, ce processus délibératif peut s'avérer plus ou moins complexe **[**Russel et Norvig, 2003**]** spécifiant ainsi le comportement de l'agent.

Discussion

Dans notre étude, nous nous intéressons à l'apprentissage centré agent et pour ce qui est de la conception d'un agent qui fonctionne en utilisant ce type d'apprentissage, il n'y a pas de solution toute faite qui fonctionne dans tous les cas, car le monde n'est jamais idéal. Il faut donc faire un travail qui porte plus sur l'analyse des contraintes liées au problème, identifier ces dernières c'est en résoudre une grande partie. De plus, à la différence des approches classiques, l'agent est directement confronté à son monde.

La mise en place d'une mesure de performance adaptée est un problème difficile : elle doit tenir compte des actions immédiates mais aussi de leurs effets à long terme. Par ailleurs, la rationalité d'un agent s'avère être un critère subjectif dépendant du concepteur. Suivant le

comportement qu'il souhaite que son agent adopte, celui-ci définira une mesure de performance plutôt qu'une autre. Il y a lieu de noter qu'un comportement rationnel d'un certain point de vue peut paraître tout à, fait irrationnel d'un autre. Nous discuterons l'impact du choix cette mesure.

L'apprentissage situé présente quelques avantages pour l'agent qui va être capable d'éviter des erreurs qu'il aurait déjà faites par le passé. Non pas parce qu'il comprend son erreur, mais parce que statistiquement l'action qui le jette dans le problème (situation) a été pénalisée donc non sélectionnée spécifiant ainsi une approche comportementale suivi par cette entité.

2.5 Approche comportementale

2.5.1 Définition et concept de l'architecture comportementale

Les Architectures comportementales [Tyrrell, 1993 ; Arkin, 1998 ; Richard, 2001] exploitent l'idée que le comportement global d'un agent peut être réalisé par la coordination de plusieurs comportements élémentaires plus simples. Un comportement élémentaire désigne un module fonctionnel autonome qui peut recevoir des informations en entrée, prendre des décisions et produire des commandes ou autres actions en sortie.
Cette méthode permet de réduire la complexité du problème global en la divisant en sous problèmes plus simples à résoudre.

Il existe deux grandes familles de techniques de coordination de comportements: les architectures compétitives et les architectures coopératives. Dans les premières, la commande effectuée est celle voulue par un unique comportement qui a été désigné par un processus de sélection. Dans les secondes, la commande finale est le résultat d'un compromis ou d'une fusion entre les commandes proposées par les différents comportements qui peuvent être actif à tous moments [Tyrrell, 1993 ; Arkin, 1998 ; Richard, 2001].

Ces comportements supposent cependant que l'environnement de l'agent soit modélisable. Si tel n'est pas le cas, il est possible d'avoir recours à des méthodes d'apprentissage permettant à l'agent d'agir dans un environnement inconnu et d'améliorer ses performances au fur et à mesure de l'acquisition de connaissances.

Comme l'affirment Russell et Norvig **[**Russel et Norvig, 2003**],** l'orientation du comportement d'un agent par des buts n'est pas suffisant pour que celui-ci ait un comportement de haute qualité. Elle permet de définir si un comportement est souhaité ou non, c'est-à-dire s'il permet d'atteindre les buts fixés ou non. Cependant, il n'est pas possible

d'établir un ordre de préférence entre les comportements souhaités. Un tel ordre de préférence peut être exprimé par une autre notion : La notion d'utilité.

Une fonction d'utilité définit des préférences sur les états. Elle permet d'associer un nombre réel décrivant le degré de satisfaction de l'agent dans chaque situation. Il est alors naturel de s'interroger sur la relation entre utilité espérée et mesure de performance.

Russell et Norvig donnent une réponse dans **[**Russel et Norvig, 2003**]**: « *if an agent maximizes a utility function that correctly reflects the performance measure by which its behavior is being judged, then it will achieve the highest possible performance score if we average over the environnements in which the agent could be placed*» . « Si un agent maximise une fonction d'utilité qui reflète correctement la mesure de performance par laquelle son comportement est évalué, alors il obtiendra les meilleurs performances possibles en réalisant une moyenne sur les environnements dans lesquels l'agent peut évoluer ».

2.5.2 Les buts multiples

Un agent immergé dans un environnement donné doit être capable d'approximer simultanément plusieurs cibles. Les buts et les actions qui y mènent, conjointement ou en deux instants différents, peuvent être antagonistes. L'agent doit alors être en mesure de distinguer les caractéristiques propres à chaque but.

Discussion

Une mesure de performance définit donc un critère global de rationalité, externe à la manière dont le comportement de l'agent sera calculé. L'utilité espérée quant à elle décrit une mesure locale de la rationalité et permet de définir un critère de sélection de l'action rationnelle et du comportement désiré.

Il nous semble maintenant possible de donner notre définition du concept d'agent, afin de situer précisément notre cadre de travail. Nous allons également énumérer les propriétés des agents que nous traiterons par la suite.

2.6 Notre définition de l'agent

Dans ce mémoire, nous nous intéressons particulièrement à la modélisation de l'interaction entre l'agent et son environnement. Notre définition donc présente des similarités avec la définition d'un agent rationnel de Russell et Norvig [Russel et Norvig, 2003] mais à laquelle nous ajoutons certaines précisions concernant les propriétés que doit vérifier l'agent.

Ces propriétés sont : l'autonomie, la réactivité au sens de Wooldridge et Jennings, la mise en situation, et la possession d'un ou de plusieurs objectifs. Nous changeons la définition introduite par Wooldridge et Jennings **[**Wooldridge et Jenning**,** 1995] pour la notion de la mise en situation, en précisant que dans notre travail, la perception et l'action sont locales. A la différence de Ferber [Ferber, 1995], notre définition n'aborde pas la dimension sociale de l'agent.

Pour résumer, la définition d'un agent que nous adoptons dans ce travail est la suivante :

Un agent est une entité située dans un environnement, capable de percevoir cet environnement et d'y agir de manière autonome et rationnelle afin d'atteindre ses objectifs.

Il doit vérifier donc les propriétés suivantes :

- ✓ La réactivité : la capacité à répondre aux événements extérieurs.
- ✓ La possession d'un ou de plusieurs objectifs.
- ✓ La situation, l'agent perçoit son environnement via ses capteurs, et peut agir en conséquence via ses effecteurs.
- ✓ L'autonomie : la capacité d'agir sans intervention d'un tiers.
- ✓ La rationalité : capacité d'effectuer l'action qui lui permet de maximiser sa mesure de performance, sur la base de ses séquence perceptives et des ses connaissances précédentes.

2.7 Problématique de Conception de l'agent intelligent

La description qui a été faite précédemment de ce qu'est notre agent le présente comme essentiellement dirigé par une détermination, un but, qui peut être traduit par une mesure de performance.

On suppose disposer de cette fonction caractéristique du problème et on cherche alors à construire un système, pour qu'à l'exécution et du fait de ses interactions avec l'environnement, le chaînage de son comportement parvient à optimiser sa fonction de performance

La conception de cet agent rationnel est dirigée par trois principaux aspects qui ont retenu notre attention, aspects que l'on peut lier à l'intelligence "naturelle" :

1. On cherche à automatiser la conception en limitant autant que possible l'intervention du concepteur. Ceci en se basant sur un principe simple : mettre au point un système guidé par ses buts qui s'adapte de manière à être intelligent c'est-à-dire de manière à être rationnel.

2. Un autre point est l'environnement dans lequel évolue l'agent et qui est le plus souvent inconnu. L'agent n'a pas de connaissances à priori donc aucune représentation prédéfinie de ce dernier n'existe.

3. Utiliser des méthodes d'apprentissage par interaction.

2.7.1 Principe de conception d'un agent rationnel

Concevoir ce système consiste à résoudre un certain nombre de problèmes [Ferber, 1997] :

- ✓ Quelle est l'architecture de l'agent, sachant que son comportement en dépend? (c.f. Chap1)
- ✓ Comment faire évoluer son comportement pour que l'agent puisse tirer parti des expériences passées et quelles en sont les conséquences sur le comportement final?
- ✓ Comment implémenter et réaliser ce système ?.

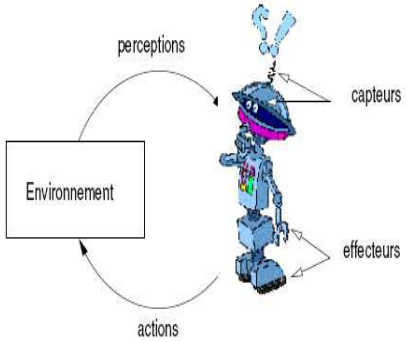
2.7.2 Point de vue adopté

Maintenant que nous avons défini l'agent et ces propriétés indissociables, ainsi que le principe de conception d'agent intelligent, il est possible d'exposer notre point de vue adopté pour la conception d'agent intelligent capable d'agir de manière autonome et rationnelle afin de résoudre la tâche qui leur a été confiée :

« La conception d'un agent intelligent consiste à définir la "**fonction objective**" ainsi que "**l'algorithme**" qui va s'attacher à l'optimiser ».

Nous pouvons alors schématiser notre principe de conception de l'agent fondé sur le principe de la rationalité.

☞ **Agent intelligent**



☞ **Entité**

- Percevant l'environnement
- Agissant sur l'environnement

☞ **Intelligence** → Autonomie

→ Rationalité

☞ **Conception**

- Description

Architecture

→ Réactif

→ Mémoire

- Comportement

→ Fonction décision

→ Fondé rationalité

Figure 2.1- Principe de conception de notre agent : Fondé Rationalité

2.7.3 Proposition de résolution du problème de conception

La notion d'intelligence est fortement liée à l'apprentissage et tous les algorithmes d'apprentissage évoqués précédemment ont certaines propriétés plus ou moins intéressantes en fonction du problème à résoudre.

Cependant, deux caractéristiques importantes se retrouvent lorsque nous évoquons la notion de conception d'agent intelligent : la représentation de l'environnement au niveau de l'agent, et la prise de décision avec représentation.

A cet effet, nous proposons d'utiliser des algorithmes d'apprentissage par renforcement (AR) pour automatiser le processus de conception d'agent réactif. Chaque agent va ainsi apprendre localement son comportement de manière à optimiser sa performance.

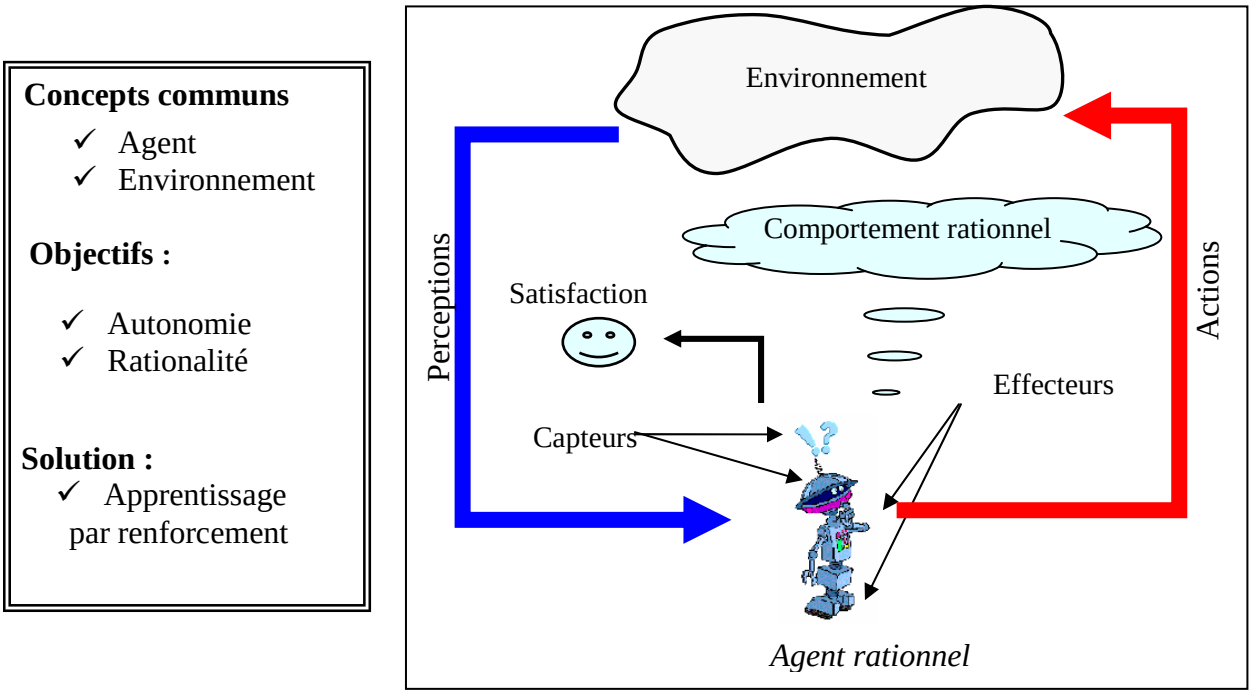


Figure 2.2 - Le schéma de fonctionnement de notre agent dans son environnement

Les avantages de cette approche sont nombreux :

✓ d'une part, un simple signal scalaire évaluant le comportement du système (la récompense) est suffisant pour apprendre. Il n'est pas nécessaire de disposer d'un professeur connaissant à l'avance la solution au problème.

✓ d'autre part, l'agent est confronté à une tâche assez simple, il est plus facile pour lui d'apprendre son comportement réactif localement que d'essayer d'apprendre globalement un comportement du système.

✓ Il offre des processus permettant à cet agent d'effectuer une synthèse de ses expériences pour mettre à jour son comportement.

✓ Les méthodes d'apprentissage par renforcement s'appuient sur le formalisme mathématique des processus décisionnels de Markov (PDM) qui offrent un outil puissant de modélisation.

2.8 Conclusion

Nous nous sommes attachés dans ce second chapitre à aborder la notion d'intelligence pour développer notre vision d'agent intelligent la quelle est basé sur le notion de rationalité. Nous avons pu donner une définition à notre concept « agent intelligent», le principe de conception adopté et ainsi que le schéma de fonctionnement. Nous terminons le chapitre en tirant profit des avantages d'une telle approche de conception.

De l’agent … au système Intelligent

Conclusion

Dans cette partie, nous avons détaillé dans le premier chapitre, une composante essentielle de l'intelligence artificielle située : la notion d'agent. Dans le second chapitre nous avons mis en évidence les notions de rationalité et d’apprentissage qui permettent de spécifier ce qu'on entend par système intelligent. Il nous a été possible de donner notre propre définition du concept d'agent afin de situer précisément notre travail et qui s’intéresse particulièrement à la modélisation de l'interaction entre l'agent et son environnement. Ceci nous permet de formaliser plus précisément le problème de conception d'agent pour en tirer des algorithmes génériques.

Pour les mêmes raisons que précédemment, nous avons dû faire des choix sur l'architecture interne des agents que nous considérerons par la suite : comme nous souhaitons disposer des agents les plus simples, nous avons décidé d’adopter des agents réactifs dotés de mémoire. Mais, l'utilisation de mémoire pose un autre problème souvent relégué au concepteur et auquel nous allons répondre dans la partie modélisation : Quel élément l'agent doit-il mémoriser et comment le représenter ?

Pour mettre en oeuvre cette idée, nous allons dans la partie suivante nous concentrer sur une catégorie de formalismes existant qui constituera notre point de départ : les processus de décision markoviens. Nous présenterons les différents algorithmes de base et l’utilisation de tels formalismes dans les différents cas. Nous montrerons dans quelle mesure ces cadres formels représentent l'interaction et la prennent en compte au niveau de l’agent pour constituer la base de ce qu’on appelle techniques d’algorithmes d’apprentissage par renforcement

Partie B

Modélisation

Partie B

Modélisation

Introduction

L'objet de cette partie est de présenter le formalisme de conception d'agent et de mettre en évidence les algorithmes que nous cherchons à mettre en œuvre. Nous allons présenter les bases de l'apprentissage par renforcement (AR) et les formalismes aidant à les décrire tels que les processus de décision de Markov (PDM). Elle se compose de cinq parties : La première introduit l'apprentissage par renforcement, ses objectifs, ses origines et son principe. La deuxième est consacrée à la résolution d'un problème par apprentissage par renforcement en présentant son formalisme mathématique. La troisième partie est dédiée aux méthodes d'optimisation de comportement. La quatrième partie présente les fonctions de généralisation pour permettre l'apprentissage dans les systèmes complexes et dans la cinquième partie, on discute des paramètres d'apprentissage et du choix de leurs valeurs d'initialisation ainsi que les différentes recherches effectuées dans ce sens. On termine par présenter les différents domaines d'application de l'apprentissage par renforcement.

Chapitre III

Apprentissage par renforcement

Chapitre III

Apprentissage par renforcement

3.1 L'apprentissage par renforcement (AR)

L'apprentissage par renforcement (AR) est un domaine qui recouvre une classe de méthodes d'apprentissage automatique à mi-chemin entre l'apprentissage supervisé et l'apprentissage non supervisé. Dans l'apprentissage supervisé, le système apprenant fournit une réponse à chaque situation présentée. Un superviseur lui montre alors la bonne réponse et le système peut ainsi corriger la sienne. Ce principe nécessite de connaître au moins une partie des réponses à apporter aux différentes situations. A l'inverse, dans l'apprentissage non supervisé, le système ne perçoit que les situations et doit apprendre des « régularités » dans celles - ci afin d'en effectuer une classification.

(Une étude bien détaillée de l'AR se trouve dans les ouvrages de R.S. Sutton [Sutton, 1988], L.P. Kaelbling [Kaelbling et* al.*, 1996] et Sutton et Barto [Sutton et* al.*, 1998]).

3.1.1 Définition

L'apprentissage par renforcement désigne toute méthode adaptative permettant de résoudre un problème de décision séquentielle. [Sutton et *al.*,1998]. Le terme "adaptatif" signifie qu'on part d'une solution inefficace, que l'agent améliore progressivement en fonction de son expérience. L'adjectif séquentiel provient du fait que la performance individuelle est évaluée le long de la trajectoire du système et non pas de manière instantanée.

3.1.2 Objectif

Le double objectif de l'apprentissage par renforcement est de conduire de manière optimale un système, au cours du temps, tout en apprenant cette conduite optimale à travers des expériences [Garcia, 1997] : c'est donc apprendre à partir d'expériences quoi faire en chaque situation, de manière à optimiser la fonction du système.

3.1.3 Origines de l'apprentissage par renforcement

Plusieurs modèles ont été à l'origine de l'apprentissage par renforcement à savoir :

3.1.3.1 Modèle biologique :

Le problème d'apprentissage par renforcement est fortement lié aux travaux effectués en biologie expérimentale dans la première moitié du XX^{ième} siècle qui consistent à analyser comment un organisme est capable de s'adapter à son environnement. Deux approches expérimentales se sont intéressées à ce problème : le conditionnement pavlovien et le conditionnement opérant.

Le conditionnement pavlovien, mis en évidence par I. Pavlov [Pavlov, 1927] modélise le processus d'apprentissage des associations entre stimuli. Il rend compte de la manière dont les animaux apprennent à associer des stimuli conditionnels à des stimuli inconditionnels : des études concernant le comportement animal ont en effet montré que lorsqu'un animal était soumis plusieurs fois à un stimulus conditionnel et que ce dernier était suivi par un second (stimulus inconditionnel), l'animal parvenait à émettre une réponse au stimulus inconditionnel dés qu'il percevait le stimulus conditionnel. Une association entre les deux stimuli était alors formée, la présentation du premier stimulus suffisait à déclancher le réflexe, alors que le second stimulus n'a même pas été présenté. Le principe du modèle reposait sur le fait que l'animal essayait de prédire le stimulus inconditionnel sur la base du stimulus conditionnel. Ce type de conditionnement ne s'intéresse pas à la formation d'associations entre plusieurs stimuli, il ne prend pas en compte les actions de l'animal.

Le conditionnement opérant [Skinner, 1938] au contraire concerne l'apprentissage de contingences entre actions et stimuli. Une contingence associe une action de l'animal à sa conséquence qui peut être un stimulus attractif ou répulsif.

Le stimulus est dit discriminant s'il permet à l'animal de déterminer dans quelle situation l'action entreprise mène effectivement à la satisfaction considérée.

Néanmoins, les deux visions précédentes se rejoignent puisque de nombreux chercheurs ont tendance à croire que les processus d'apprentissage impliqués dans le conditionnement pavlovien et le conditionnement opérant sont identiques.

On lie aussi à posteriori l'apprentissage par renforcement à l'apprentissage par essai-erreur d'E. Thorndike [Thorndike, 1911] plus proche des expériences de conditionnement opérant : lors d'un apprentissage essai - erreur, les actions dont les conséquences ont été bénéfiques auront tendance à être réémises de manière plus fréquente dans le futur alors que les actions ayant eu des conséquences néfastes auront au contraire tendance à être exprimées moins souvent

3.1.3.2 Modèle psychologique

L'apprentissage par renforcement est inspiré aussi du modèle de Rescorla et Wagner [Rescorla et *al.*, 1972] en psychologie plus précisément du '' Behaviorisme'' [Rouanet, 1964] . Ce modèle est développé parallèlement aux recherches sur le conditionnement de Pavlov et rend compte de l'apprentissage des associations dans le conditionnement classique. Il décrit de manière simple comment et à quelle vitesse les stimuli conditionnels sont renforcés. D'après Rescorla-Wagner, une valeur V(sc) est associée à chaque stimulus conditionnel (sc) et à chaque essai, la valeur associative des n stimuli conditionnels présents est mise à jour par l'équation suivante :

$$V(sc_i) = V(sc_i) + \beta(R(si) - \sum_{i=1}^{n} V(sc_i)) \qquad (3.1)$$

où R(si) : désigne l'intensité du stimulus inconditionnel et β représente la vitesse d'apprentissage qui peut varier selon les espèces animales. Ce modèle a été d'un grand apport pour l'apprentissage par renforcement parce qu'il rend compte d'un grand nombre de phénomènes observés [Gérard, 2002]. Il a été repris et adapté dans [Sutton et al. 1998] de manière à proposer des algorithmes d'apprentissage par renforcement.

3.1.3.3 Modèle Neuroscience

Une autre règle d'apprentissage célèbre est la loi de Hebb [Hebb. 1949], laquelle a conduit au modèle de réseau de neurones présenté par Hopfield [Hopfield, 1982]. Selon cette loi, la force de la connexion (synaptique) présente entre deux neurones est fortement augmentée, en d'autres termes, se trouve ''renforcée'' si ces neurones sont simultanément actifs : on parle de renforcement synaptique.

Ce renforcement synaptique peut être positif, augmentant ainsi la force de la connexion, on parle alors de facilitation de la transmission synaptique (ou potentialisation). Au contraire, si ce renforcement est négatif, on parle alors d'inhibition synaptique (ou dépression).

Toutefois, les réseaux connexionnistes n'ont pas pour objectif de conduire des raisonnements sur des algorithmes d'apprentissage par renforcement ou de servir de support à des démonstrations mathématiques au sujet de ces derniers.
Cherchant le cadre de travail approprié, on préfère l'utilisation de modèles mathématiques présentés ci-dessous et qui sont plus spécifiquement adaptés à de tels objectifs.

3.1.3.4 Modèles mathématiques

On n'essaiera pas de reconstruire un historique complet des avancées dans le domaine des mathématiques qui ont donné naissance aux outils récents les plus efficaces pour travailler sur l'apprentissage par renforcement. On se contentera de citer :

- ✓ D'abord la théorie des probabilités, outil capable de représenter et gérer l'incertitude, qui procure le cadre de travail adéquat pour décrire un monde, ses lois d'évolution et les croyances incomplètes sur ce dernier ;
- ✓ Puis les chaînes de Markov, formalisme au sein de la théorie des probabilités permettant d'étudier l'évolution des états d'un système en connaissant les lois régissant le passage d'un état vers un autre ;
- ✓ La théorie de l'utilité [Von Neumann et *al*, 1944], fournit aussi un cadre pour rendre cohérents des ensembles de préférences et apporte la notion d'évaluation d'une décision; notion que l'on va pouvoir mettre en correspondance avec la notion de performance ;
- ✓ Et plus précisément la théorie de la décision, issue de la recherche opérationnelle, qui s'intéresse à un problème fondamental pour un agent : celui de la prise de décision. Elle se fonde sur les axiomes de probabilité et d'utilité. Elle propose des éléments syntaxiques permettant de formaliser des problèmes de prises de décision en environnement incertain et offre des outils permettant de calculer les réponses à ces problèmes sous la forme de fonctions comportementales.

Ces différentes théories regroupent les aspects utiles à une formalisation complète de l'apprentissage par renforcement [Horvitz et *al.*, 1988] .

Finalement, on peut conclure que l'apprentissage par renforcement constitue réellement un domaine pluridisciplinaire (Figure 3.1).

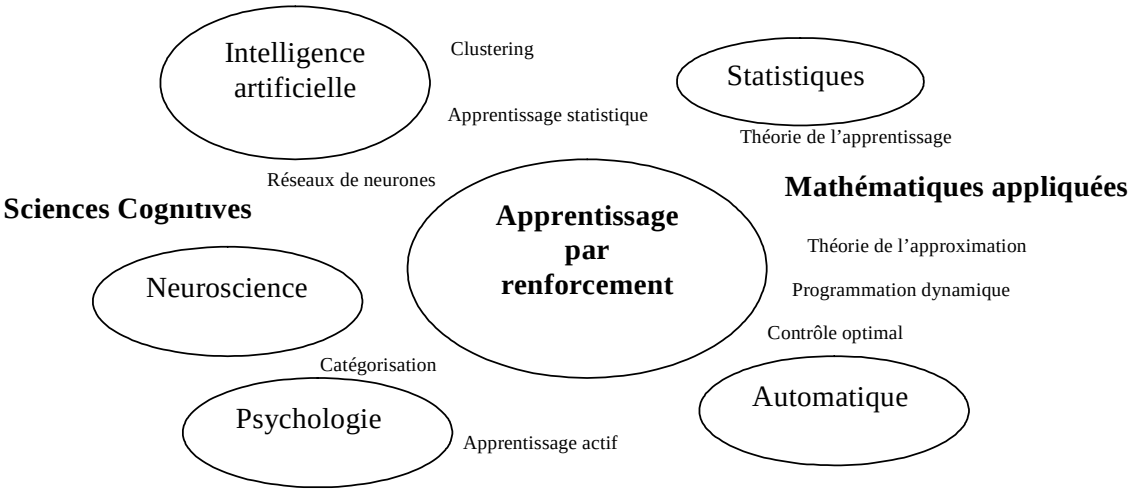


Figure 3.1 - L'apprentissage par renforcement : domaine pluridisciplinaire [Munos, 2007]

3.1.4 Principe de l'apprentissage par renforcement

D'après la définition du S. Sutton et G. Barto [Sutton et al. 1998] et J. Baxter [Baxter, 1999], l'apprentissage par renforcement définit un type d'interaction entre l'agent et l'environnement (figure 3.2).

Son principe est le suivant : depuis une situation réelle «s» dans l'environnement, l'agent choisit et exécute une action « a » qui provoque une transition vers l'état « s' ». Il reçoit en retour un signal de renforcement « r » négatif de type pénalité si l'action conduit à un échec ou positif de type récompense si l'action est bénéfique ; enfin un signal nul signifie une incapacité à attribuer une pénalité ou une récompense.

A chaque modification de situation, le système reçoit une note ou renforcement qui qualifie ce changement. L'agent utilise alors ce signal pour améliorer sa stratégie, c'est à dire la séquence de ses actions, afin de maximiser le cumul de ses récompenses futures en choisissant les actions qui lui apporteraient plus de récompenses et moins de punitions.

Un paradigme classique pour présenter ce problème est celui de l'agent autonome plongé au sein d'un environnement et qui recherche à travers des expériences itérées au sein de cet environnement un comportement décisionnel optimal, associant à l'état courant l'action courante à exécuter [Sigaud, 2004].

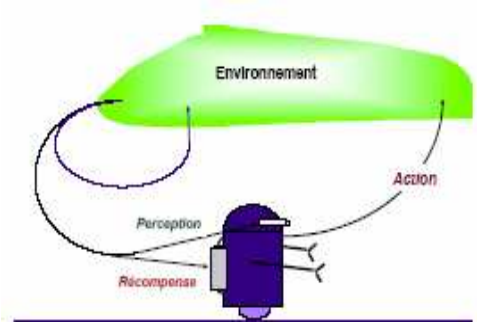


Figure 3.2 - Apprentissage par renforcement : Interaction agent / environnement.

3.1.5 Paramètre d'influence :

Les éléments à prendre en compte dans l'apprentissage par renforcement sont multiples [Buffet, 2000 ; Buffet, 2003] :

a. Temps

L'espace de temps a des formes différentes, il peut être :

- Discret ou continu.
- Fini ou infini.
- Déterminé ou aléatoire.

La plupart des études sur l'apprentissage par renforcement utilisent un espace de temps discret.

b. Etats

L'ensemble « S » appelé états caractérise les situations d'un agent dans l'environnement à chaque instant, elles peuvent se présenter sous trois formes :

- Une situation relationnelle de l'agent par rapport à l'environnement (position).
- Une situation propre à l'environnement (modifications du milieu).
- Une situation interne à l'agent (sa mémoire, ses capteurs, etc.).

Les trois formes d'état peuvent être présentes en même temps en fonction du problème traité.

c. Actions

Un agent peut choisir d'accomplir, en vue de modifier l'état de son environnement, une action parmi celles possibles à chaque instant « t », elle peut être instantanée ou durer jusqu'au prochain instant. A chaque état de l'espace d'état est associé un ensemble d'actions possibles de l'espace d'action,

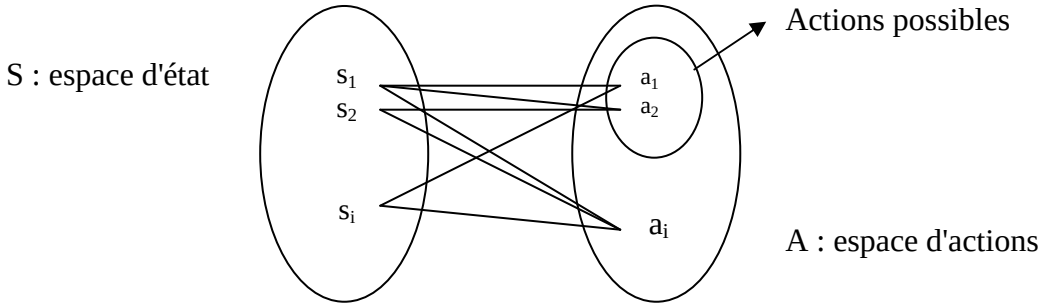


Figure 3.3 - Relation états, actions.

d. Environnement

Russell et Norvig [Russel et Norvig, 2003] ont identifié un certain nombre de propriétés permettant de caractériser l'environnement d'un agent et qui sont utiles pour choisir une méthode d'apprentissage.

- Accessible / inaccessible: l'agent a-t-il accès à l'état complet de l'environnement ou certaines informations restent-elles non ou mal connues?

- Déterministe / non-déterministe : le prochain état de l'environnement est-il complètement déterminé par son état courant et l'action sélectionnée par l'agent?

- Episodique / non-épisodique : on parle d'environnement épisodique si l'expérience de l'agent peut être divisée en épisodes. Chaque épisode consiste en une phase de perception puis en une phase d'action dont le résultat ne dépend que de la période courante.

- Statique / dynamique : un environnement est dit dynamique s'il peut changer pendant la prise de décision de l'agent. Il est dit statique dans le cas contraire.

- Discret / continu: s'il existe un nombre limité de perceptions et d'actions possibles, on parlera d'environnement discret.

e- Fonction de récompense

La fonction de récompense « r » décrit les motivations individuelles de l'agent. Lorsque l'agent se trouve dans un état « s » et qu'il effectue une action « a », il reçoit une récompense immédiate R(s, a). Cette récompense peut être perçue comme venant de l'environnement ou comme un jugement interne propre à l'agent.

Ce deuxième point de vue qui est à prendre de préférence, puisqu'il correspond à l'idée qu'un point essentiel dans la conception d'un agent est la définition de ses objectifs. La rationalité d'un agent s'exprime sous la forme d'un critère de performance individuel lié à cette fonction de récompense.

Plusieurs critères de performance individuelle peuvent être définis. Ces critères dépendent de la catégorie de problèmes que l'on cherche à résoudre :

- ✓ S'il s'agit d'un problème en horizon fini, c'est à dire pour lequel l'agent évolue pendant un nombre de pas de temps fixe à «n » connu à priori, un critère de performance possible est la somme des récompenses reçues le long de la trajectoire de l'agent : $E[\sum_{t=0}^{n} r_t]$
- ✓ S'il s'agit d'un problème en horizon infini, c'est à dire pour lequel le processus évolue indéfiniment, d'autres critères de performance doivent être définis. Parmi les critères habituellement utilisés, on peut citer :

- le critère moyen : $\frac{1}{n}.E[\sum_{t=0}^{n} r_t]$
- le critère gamma pondéré : $E[\sum_{t=0}^{\infty} \gamma^t r_t]$ avec $\gamma \in [0,1[$

Le paramètre γ appelé « facteur de décompte » peut être compris comme un paramètre permettant de déterminer l'importance du futur dans la prise de décision.

3.2 Résolution du problème de l'apprentissage par renforcement

3.2.1 Problématique

Un agent a été présenté comme étant une entité intelligente en ce sens qu'elle interagit avec son environnement à travers ses perceptions et actions, et s'adapte de manière à atteindre un objectif donné. Cet agent apprend aussi de façon individuelle son comportement, déterminé par ses propres expériences.

La description proposée de l'agent comme étant dirigé par une détermination à accomplir un but et peut être traduite par une mesure de performance. Celle-ci se déduit d'un signal de renforcement, récompense évaluée de manière numérique et que l'agent cherche à optimiser à long terme.

Une première méthode de résolution est de planifier le comportement de l'agent à l'avance, connaissant un modèle suffisamment complet de l'environnement. Cependant, lorsque l'environnement est mal connu, il est souvent très difficile d'en concevoir un modèle réaliste et très fastidieux ou même impossible de définir le comportement de l'agent à la conception. Dans ce cas, la planification n'est plus utilisable et il faudra apprendre par essai - erreur un comportement qui va résoudre le problème posé à l'agent.

3.2.2 Principe de résolution

Le point de départ est l'utilisation de chaînes de Markov. Cet outil de la théorie des probabilités permet d'étudier l'évolution des états d'un système en connaissant les lois qui régissent les transitions du système, en l'occurrence la probabilité de passer d'un état « s » à un autre état « s' ». L'idée est aussi de faire un lien entre les chaînes de Markov et la théorie de l'utilité. Il s'agit d'une part de définir des valeurs estimant la "qualité" d'une transition, et d'autre part de permettre un contrôle partiel sur ces transitions grâce à ce qu'on appelle « actions » : ce ne sont autres que les processus décisionnels de Markov (PDM) qui pourraient formaliser un tel système.

Le PDM est donc l'un des outils qui permet de décrire le comportement dynamique de l'agent et fournit ainsi le cadre mathématique adapté pour l'apprentissage par renforcement dans le cas déterministe [Putterman, 1994 ; Sutton et al., 1998 ; Fabiani et al, 2001]. Il s'agit là d'une classe de modèles de Markov, eux-mêmes appartenant aux processus stochastiques définis ci-dessous:

3.2.2.1 Les processus stochastiques

Un processus stochastique représente une évolution, généralement dans le temps, d'une variable aléatoire. C'est une famille de variables aléatoires X(t) où t représente le temps.

3.2.2.2 Suite stochastique

Soit un système pouvant se trouver dans un ensemble dénombrable d'états $S=\{s_1,s_2,...,s_N\}$. Entre les instants « t » et « $t + 1$ », le système passe aléatoirement de l'état « s_i »à l'état « s_j ».

Pour représenter ce processus aléatoire, on définit la variable « q_t » de la façon suivante: $q_t=s_i$ signifie que le système est dans l'état « s_i » au temps « t ». Par définition l'ensemble $(q_1,q_2,...,q_t,\ldots)$ est une suite stochastique à ensemble discret d'états.

3.2.2.3 Chaîne de Markov

Une suite stochastique vérifie la propriété de Markov si l'état du système à un instant donné dépend uniquement de l'état précédent. Enfin, une chaîne de Markov est dite stationnaire si la probabilité de la transition entre états est indépendante du temps.

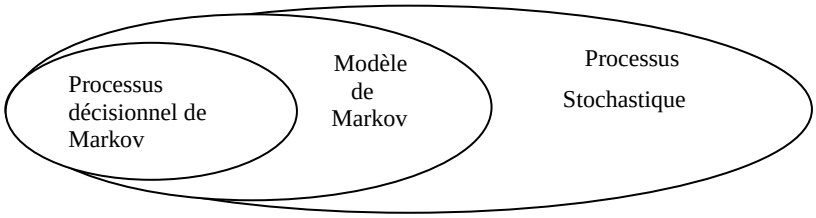


Figure 3.4 - ***Structure des Processus stochastiques.***

3.2.3 Formalisme du Processus décisionnel de Markov (PDM) :

Un PDM est défini par un tuple < *S; A; T; R* >.

- *S* désigne l'ensemble d'états discrets ;
- *A* désigne l'ensemble d'actions discrètes ;
- $T : S \times A \times S \rightarrow [0; 1]$ est une fonction appelée matrice de transition ;
- $R : S \times A \rightarrow \mathrm{R}$ est une fonction appelée fonction de récompense.

S décrit le système, A, fournit l'ensemble des actions possibles, et la fonction de récompense R permet de définir les objectifs de l'agent. La matrice de transition T représente la dynamique du système et son évolution. Elle caractérise les réactions de l'environnement aux actions émises par l'agent.

Un processus de décision markovien (PDM) vérifie la propriété de Markov. Celle-ci stipule que la probabilité d'atteindre un état « s » à la temps « t+1 » dépend uniquement de l'état « s » du système au temps « t »et de l'action « a » exécutée au temps « t » :

$$P(s_{t+1} \mid ,a_t) = P(s_{t+1} \mid s_0, a_0, s_1, a_1, .. s_t, a_t) = P(s_{t+1} \mid s_t, a_t) = T(s_t, a_t, s_{t+1}) \qquad (3.2)$$

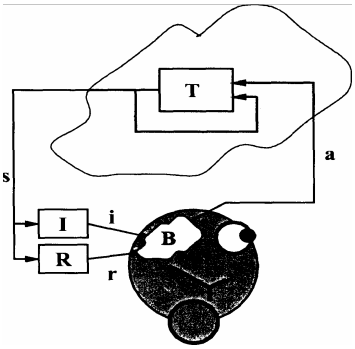
3.2.4 Optimalité dans les processus décisionnels de Markov (PDM) :

Nous avons présenté le formalisme des processus décisionnels de Markov. Ce dernier permet de modéliser des problèmes décisionnels dans un environnement stochastique. Nous allons maintenant nous intéresser à la résolution d'un tel problème :

Résoudre un processus décisionnel de Markov (PDM) consiste alors à estimer le comportement d'un agent étant donné:

- ✓ un ensemble d'actions possibles,
- ✓ une fonction de performance individuelle,
- ✓ un état de départ
- ✓ et les lois du monde (pas forcément connues)

La figure ci-dessous montre un agent dans son environnement, ainsi que les différentes grandeurs utiles.



a : action
s : état
i: entrée (perception)
r : renforcement
T : fonction de transition
(loi régissant l'environnement)
B : comportement

Figure 3.5 - Schéma de principe d'un agent dans son environnement [Buffet, 2000]

Considérons un PDM *(S, A, T, R),* nous allons définir ce qu'est le comportement de l'agent et la valeur attribuée. Pour montrer ensuite qu'il existe au moins un comportement optimal.

3.2.4.1 Comportement et Politique

Le comportement de l'agent représente la manière dont il va choisir une action dans un état donné. Ce comportement est défini par une politique π, qui est une fonction de l'ensemble des couples état-action dans l'intervalle [0,1] ; $\pi : S \times A \rightarrow [0,1]$ avec la contrainte suivante : $\forall s \in S. \ \Sigma_{a \in A} \pi(s,a)=1$. Une telle politique est dite stochastique. En effet, pour chaque état, π nous donne une probabilité de choisir une action donnée.

3.2.4.2 Exécution d'une politique

Pour un PDM donné, l'exécution d'une politique π correspond à la boucle perception - action d'un agent et est constituée de cycles de quatre étapes:

- ✓ l'agent observe l'état « s » de l'environnement,
- ✓ il détermine l'action « a » qu'il souhaite entreprendre en fonction de sa politique π, $a \leftarrow \pi(s)$,
- ✓ il effectue cette action et l'état du monde se trouve modifié T(s, a, s'),
- ✓ il reçoit une récompense $r \leftarrow R(s, a)$.

3.2.4.3 Valeur d'une politique

Dans la mesure où la réalisation de certaines actions dans certains états procure à l'agent une récompense, on peut associer à chaque politique une valeur.

La fonction de valeur associée à une politique π indique pour tout état « s_t » la quantité de récompense que l'agent peut espérer recevoir sur le long terme s'il suit la politique π à partir de l'état «s_t». Cette quantité se décompose en une récompense reçue immédiatement pour avoir accompli l'action stipulée par π dans l'état « s_t» et des autres récompenses reçues ultérieurement en continuant à se comporter conformément à cette politique.

Il existe deux types de fonctions de valeur $V(s_t)$ et $Q(s_t,a)$ représentant respectivement la fonction de valeur d'un état s_t et la fonction de valeur d'un couple état-action pour une politique donnée.

3.2.4.3.1 Valeur d'un état pour une politique donnée

La valeur d'un état « s » $\in S$ pour une politique π donnée, se note $V^{\pi}(s)$; telle que $V^{\pi} : S \rightarrow R$, nous utilisons le modèle à horizon infini et renforcements actualisés. On a alors :

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid st = s\right\} \quad \text{avec } 0 \leq \gamma < 1. \qquad (3.3)$$

Cette fonction est l'espérance des sommes actualisées des renforcements lorsque l'on part de s et que l'on suit la politique π par la suite.

Pour un environnement déterministe cette fonction s'écrit simplement :

$$V^{\pi}(s) = \left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid st = s\right\} \quad \textit{avec } 0 \leq \gamma < 1. \qquad \textit{(3.3)}$$

3.2.4.3.2 Valeur d'un couple état-action pour une politique donnée

Nous allons décrire l'autre fonction de valeur représentant la valeur d'un couple état-action. Cette fonction est primordiale pour les algorithmes d'apprentissage.

La valeur d'un couple état-action *(s, a)* pour une politique π donnée, se note $Q^{\pi}(s,a)$, et définie par $Q^{\pi} : S \; X \; A \rightarrow R$.

Pour un modèle à horizon infini et renforcements actualisés, cette fonction Q^{π} s'écrit alors :

$$Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \;\middle|\; s_t = s, a_t = a\right\} \quad \text{avec } 0 \le \gamma < 1 \qquad (3.4)$$

C'est l'espérance des sommes actualisées des renforcements reçus lorsque l'on part de « s » et que l'on exécute « *a* » en « *s* »et que l'on suit la politique π par la suite.
Pour un environnement déterministe cette fonction s'écrit simplement :

$$Q^{\pi}(s,a) = \left(\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \;\middle|\; s_t = s, a_t = a\right) \qquad \text{avec } 0 \le \gamma < 1 \qquad (3.5)$$

3.2.4.4 Équations de Bellman :

Les fonctions de valeurs V^{π} et Q^{π} pour une politique π donnée vérifient une relation fondamentale appelée équation de Bellman [Bellman, 1957]. Pour tout état $s \in S$ on a:

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \;\middle|\; st = s\right\}$$

$$= E_{\pi}\left\{r_{t+1} + \sum_{k=1}^{\infty}\gamma^{k} r_{t+k+1} \;\middle|\; st = s\right\}$$

$$= E_{\pi}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \;\middle|\; st = s\right\}$$

$$= \sum_{s'\in S} T(s,\pi(s),s')\left[R(s,\pi(s),s') + \gamma E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \;\middle|\; s_{t+1} = s'\right\}\right]$$

$$= \sum_{s'\in S} T(s,\pi(s),s')\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right]$$

D'où

$$V^{\pi}(s) = \sum_{s'\in S} T(s,\pi(s),s')\left[R(s,\pi(s),s') + \gamma V^{\pi}(s')\right] \qquad (3.6)$$

Similairement, pour la fonction Q^{π} et pour tout s$\in$ S et tout a $\in$ A on a les équations suivantes :

$$Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \;\middle|\; s_t = s, a_t = a\right\}$$

$$= E_{\pi}\left\{r_{t+1} + \sum_{k=1}^{\infty}\gamma^{k} r_{t+k+1} \;\middle|\; s_t = s, a_t = a\right\}$$

$$= E_{\pi}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+2} \;\middle|\; s_t = s, a_t = a\right\}$$

$$= \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \ \middle| s_{t+1} = s' \right\} \right]$$

$$= \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma V^\pi(s') \right]$$

D'où

$$Q^\pi(s,a) = \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma V^\pi(s') \right] \qquad (3,7)$$

D'après les deux équations (3,6) et (3,7), nous remarquons qu'il existe une relation entre V^π et Q^π : $V^\pi(s) = \sum_{s' \in S} T(s,\pi(s),s') \left[R(s,\pi(s),s') + \gamma V^\pi(s') \right]$

Et donc $V^\pi(s) = Q^\pi(s, \pi(s))$ (3.8)

3.2.4.4.1 Recherche de la politique optimale

Un principe souvent utilisé pour la résolution des problèmes d'optimisation à plusieurs étapes de décision est celui énoncé par Bellman en 1957 repris par Sutton [Sutton et *al*. 1998]: une suite optimale de commandes a la propriété que, quelle que soit l'étape, les commandes suivantes doivent constituer une suite optimale de décisions pour la suite du problème [Bellman, 1957].

Définition 1: soient $\pi 1$ et $\pi 2$ deux politiques déterministes, définies sur le PDM. On définit la relation $\leq$ telle que $\pi 1 \leq \pi 2$, si et seulement si, $\forall s \in S, V^{\pi 1}(s) \leq V^{\pi 2}$

Définition 2 : Une politique π est dite optimale si et seulement si, pour toutes politiques π': $\forall s \in S, V^{\pi'}(s) \leq V^{\pi}(s)$. Une telle politique sera notée π^*.

La recherche de la politique optimale suppose à priori deux phases : une phase d'estimation de la politique courante et une phase d'amélioration de cette politique. Il est néanmoins possible d'entre mêler ces deux phases selon le principe de l'itération de stratégie généralisée (ISG) qui est présenté par le schéma ci-dessous.

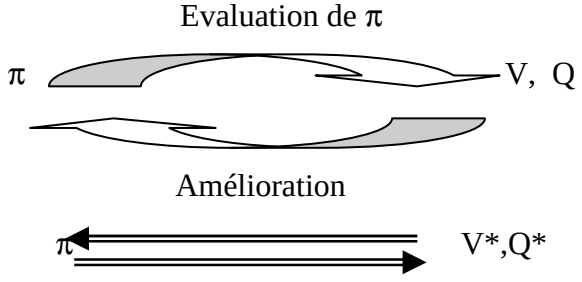


Figure 3.6 - Principe de l'Itération de Stratégie Généralisée (ISG) [Sutton et al. 1998]

3.2.4.4.2 Calcul de la valeur optimale d'une politique

Bellman [Bellman, 1957] a montré que la fonction de valeur d'une politique peut être calculée par récurrence grâce à l'équation (3,6) et (3.7)

Pour les deux fonctions de valeur de la politique V*et Q*, les équations d'optimalité de Bellman sont :

$$V^*(s) = \max_{a \in A(s)} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right] \quad (3.9)$$

$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s')\left[R(s,a,s') + \gamma \max_{a' \in A(s)} Q^*(s,a')\right] \quad (3.10)$$

Discussion

Un système d'équations peut être déduit alors en appliquant l'équation de Bellman à chaque état. Il a été montré que ce système admettait une solution unique à partir de laquelle une politique optimale peut être déduite [Sutton et al. 1998].

Bien que l'équation de Bellman admette une et une seule solution V* (respectivement Q*), il peut exister plusieurs politiques optimales si celles-ci ont toutes pour valeur V* (respectivement Q*).

3.3 Algorithmes d'optimisation de comportement

Jusqu'à présent, nous avons défini formellement un processus de décision markovien (PDM) et nous avons posé les équations définissant la valeur d'une politique quelconque et la valeur d'une politique optimale.

Nous nous penchons à présent sur les algorithmes qui permettent à un agent de découvrir une politique optimale lorsqu'il est confronté à un PDM.

Il existe trois classes d'algorithmes qui remplissent cette fonction :

- ✓ Méthodes de Programmation Dynamique (PD)
- ✓ Méthodes de Monte-Carlo (MC)
- ✓ Méthodes des Différences Temporelles (TD)

Nous allons décrire brièvement les deux premières méthodes car elles nous servent de bases pour présenter la suite. Nous consacrerons une section plus détaillée aux méthodes de différence temporelle qui constituent le cadre de l'apprentissage par renforcement proprement dit.

3.3.1 Méthode de la Programmation Dynamique (PD)

La programmation dynamique [Bellman, 1957] recouvre un ensemble d'algorithmes qui permettent de trouver la politique optimale sur un problème de décision markovien, dans le cas où la fonction de transition et la fonction de récompense (T et R) sont connues. On dit que ces méthodes permettent de résoudre le problème de planification et non d'apprentissage.

Afin de présenter le plus clairement possible l'ensemble des algorithmes relevant de la programmation dynamique dans un cadre unifié, on distingue deux étapes élémentaire de ces algorithmes : l'évaluation d'une politique et l'amélioration de celle-ci.
Tous les algorithmes de programmation dynamique résultent de diverses combinaisons de ces deux étapes. [Sutton et al. 1998].

3.3.1.1 Principe de l'évaluation et de l'amélioration de la politique

Soit une politique, on appelle évaluation de cette politique le processus qui consiste à déterminer la valeur de chacun des états pour un agent qui se conforme à cette politique en appliquant, le principe de Bellman [Bellman, 1957]. Une fois la politique π est estimée suite au critère de convergence, l'amélioration de cette politique consiste à choisir les actions qui privilégient les transitions vers les états à valeur associée maximale. Ce principe de Bellman est à la base de plusieurs algorithmes de résolution, en particulier Value Itération et Policy Itération.

3.3.1.2 Algorithme d'itération de la politique (Policy Itération)

L'idée de l'algorithme d'itération des politiques est d'évaluer une certaine politique et, en fonction de cette évaluation, de calculer une nouvelle politique qui est meilleure que l'ancienne et ainsi de suite. Il est constitué donc d'une succession d'évaluations de la politique courante suivie d'une amélioration de cette politique, reposant sur le résultat des ces dernières : c'est le principe de l'itération de stratégie généralisée (ISG) vu précédemment.

Les étapes de cet algorithme sont les suivantes :

1- démarrer avec une politique π quelconque.

2- calculer sa fonction de valeur.

3- améliorer la politique.

4- retourner à 2- si la politique a été améliorée.

3.3.1.3 Algorithme d'itération de la valeur (Value Itération)

Dans cet algorithme, l'amélioration de la politique est faite au fur et à mesure du calcul de sa fonction de valeur. Cette méthode est fortement adoptée.

Cet algorithme est constitué des étapes suivantes:

1- Démarrer avec une politique π quelconque.

2- Calculer $V^{\pi}{}_{n}$ selon l'algorithme de l'évaluation de la politique.

3- Améliorer π selon $V^{\pi}{}_{n}$

4- Retourner à 2- si π a été améliorée.

Discussion

✓ La programmation dynamique est d'utilisation limitée parce qu'elle suppose connu un modèle de l'agent dans son environnement qui lui permet de dériver une politique optimale en itérant sur ses données. Il n'y a aucun gain d'information dans un tel processus.

✓ Elle est seulement utile dans le cas où la dynamique de l'environnement ne change pas beaucoup, car elle présente l'avantage d'être incrémentale (on peut réaliser les itérations successives qui convergent peu à peu vers la fonction de valeur optimale).

✓ Elle pose cependant des problèmes de temps de calcul.

Dans l'apprentissage sans un modèle du monde, l'agent ne connaît pas la fonction de récompense R et la fonction de transition T donc l'apprentissage se fait d'après l'expérience. On distingue ici les deux méthodes « Monté Carlo » et « Différence temporelle »

3.3.2 Méthode de Monte Carlo (MC)

Les méthodes de Monte-Carlo utilisées pour l'apprentissage par renforcement ont été explicitement identifiées récemment dans les années 90 [Rubinstein, 1981 ; Eli et *al.*, 2001]

3.3.2.1 Principe de MC

On ne s'intéresse ici qu'à des tâches épisodiques. L'agent réalise une série d'expériences et à chaque expérience réalisée, il mémorise les transitions qu'il a effectuées et les récompenses qu'il a reçues. Disposant d'une trajectoire et des retours associés à chaque transition, on peut observer les retours suivant le passage par chaque état de la trajectoire. On nomme « trajectoire » la suite d'états parcourus depuis un état initial à un état final.

Si l'on dispose d'un grand nombre de passages par cet état et d'un grand nombre de trajectoires, on met alors à jour une estimation de la valeur des états (la moyenne des retours observés pour chaque état). Au fil de plusieurs expériences, la valeur estimée associée

à chaque état converge alors vers la valeur exacte de l'état pour la politique qu'il suit. Il faut que tous les états soient visités un grand nombre de fois pour obtenir une approximation satisfaisante de V^{π}.

Pour estimer la fonction V^{π} d'un état « s » selon la politique π, il faut estimer R_t qui est la moyenne des gains obtenus après n visites :

$$V^{\pi}(s) = \frac{1}{n}\sum_{n} R_t^n \quad \text{avec} \quad R_t^n = r_{t+1}^n + \gamma . r_{t+2}^n + \gamma^2 r_{t+3}^n + ... + \gamma^T r_T^n \qquad (3.11)$$

L'algorithme utilisé est constitué des étapes suivantes:

1- Générer une trajectoire (une séquence (S, a)) selon la politique π

2- Estimer les valeurs des $V^{\pi}(s_i)$ selon les observations effectuées sur la trajectoire.

3- Améliorer la politique π et retourner à l'étape 1.

Discussion

- ✓ Les méthodes Monté Carlo ne présupposent aucune connaissance à priori d'un modèle mais elles ne sont pas incrémentales.
- ✓ Les méthodes de Monte-Carlo utilisent seulement les résultats de l'interaction réelle ou simulée avec un environnement.
- ✓ L'apport principal de ces méthodes réside dans la manière d'estimer la valeur d'un état sur la base de la réception de valeurs successives de récompense cumulée associée à cet état lors de plusieurs trajectoires distinctes.
- ✓ Ces méthodes sont peu utilisées en pratique, car leur fonctionnement exige que l'apprentissage soit décomposé en une succession d'épisodes de longueur finie, faute de quoi la mise à jour de l'estimation de la valeur des états ne peut avoir lieu [Sigaud, 2004].
- ✓ Leurs propriétés de convergence ne sont pas encore claires et leur efficacité a été en pratique peu évaluée [Sutton et al. 1998].
- ✓ Dans ces méthodes, le temps n'est pas une variable explicite.

Nous allons présenter maintenant les méthodes de différence temporelle, qui sont les plus utilisées dans la communauté de l'apprentissage par renforcement.

3.3.3 Les méthodes différences temporelles (TD)

Les méthodes TD ('Temporal Différence learning') sont des combinaisons des idées des méthodes « Monté Carlo » et « Programmation Dynamique » et ont été développées par R. Sutton [Sutton, 1988]. Les méthodes Monte-Carlo servent à estimer les fonctions valeurs sans s'appuyer sur un modèle mais sur une phase d'expérimentation constituée d'un grand nombre d'épisodes. De même, les méthodes différences temporelles peuvent apprendre à partir de l'expérience sans avoir besoin d'un modèle de l'environnement et sans attendre la fin de chaque épisode. Comme les méthodes de la programmation dynamique, les méthodes « TD » calculent pendant l'épisode, les nouvelles évaluations à partir des évaluations précédentes.

3.3.3.1 Principe

Précédemment, dans la méthode de Monte Carlo, nous avons calculé les valeurs des états de la manière suivante (3.11):

$$V_{k+1}(s) = \frac{1}{k+1}\sum_{n=1}^{k+1} R_n = \frac{1}{k+1}\sum_{n=1}^{k} R_n + R_{k+1}$$

$$= \frac{1}{k+1}(kV_k(s_i) + R_{k+1} + V_k(s_i) - V_k(s_i)) = V_k(s_i) + \frac{1}{k+1}(R_{k+1} - V_k(s_i))$$

On aura $V_{k+1}(s_i) = V_k(s_i) + \alpha_{k+1}(R_{k+1} - V_k(S_i))$ (3.12)

Nouvelle Estimation = Ancienne Estimation + Facteur itération * [Gain du parcours - Ancienne Estimation]

Soit « s_t » un état non terminal visité à l'instant t, la mise à jour de la fonction V^{π} est menée sur la base de ce qui arrive après cette visite, la cible : R_{t+1}. En conséquence:

$$R_{t+1} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\} = E_{\pi}\left\{r_t + \gamma V_k^{\pi}(s_{t+1}) \mid s_t = s\right\}$$

et donc $V_{k+1}(s_t) = V_k(s_t) + \alpha[\, r_t + \gamma V_k^{\pi}(s_{t+1}) - V_k(s_t)]$ (3.12.a)

La différence : $r_t + \gamma V_k^{\pi}(s_{t+1}) - V_k(s_t)$ est appelée erreur de prédiction « Différence temporelle » est : Si nous utilisons la fonction de valeur « état-action » on aura la fonction suivante pour une itération «i» :

$$Q^{\pi}_{k+1}(s_i, a_i) = Q^{\pi}_k(s_i, a_i) + \alpha_k(r_i + \gamma Q^{\pi}_k(s_{i+1}, a_{i+1}) - Q^{\pi}_k(s_i, a_i)) \quad (3.13)$$

Discussion

✓ Dans les méthodes TD, l'évaluation est faite à un pas de temps sans recours à un modèle. Ces méthodes ont conduit à plusieurs algorithmes tels : Sarsa, AHC-learning, Q-learning, R-learning, critique- acteur, TD(λ), Sarsa(λ), Q-leaming(λ). Ces approches se distinguent par le type de fonction valeur estimée, et par les techniques d'évaluation et d'amélioration des stratégies (voir Annexe I).

En se référant à [Suton et *al.*, 1998], nous terminons cette partie en dressant un tableau comparatif des méthodes d'apprentissage par renforcement précédemment citées:

	Méthodes DP Programmation dynamique	**Méthodes MC Monté Carlo**	**Méthodes TD Différence temporelle**
II faut connaître le modèle du processus	Oui (π(s,a),P R,) à connaître	Non	Non
Le calcul implique une valeur estimée	Oui	Non	Oui
Profondeur de l'espace de recherche	Scrute toutes les actions possibles à partir d'un état et actualise à partir des valeurs estimées des états suivants	Déroule un épisode complet pour apprendre	Actualise à partir de du renforcement immédiat et de la valeur estimée de l'état suivant
Preuves de convergence	Oui	Convergence vers π* non prouvée	Oui
Complexité de calcul	Temps de calcul polynomial avec le nombre d'états	Estimation plus précise si grand nombre d'épisodes.	Calcul simple, sur tous les états un grand nombre de fois
Avantages	Théoriquement important Planification (optimisation, commande optimale)	Basé seulement sur les résultats de l'interaction réelle ou simulée avec un environnement. Pas besoin de scruter tous les états.	Apprentissage sans attendre la fin de l'épisode. Conduit à de nombreux algorithmes d'apprentissages.
Inconvénients	Limité à des problèmes de quelques millions d'états.	Efficacité en pratique peu évaluée Seulement pour des tâches épisodiques	Limitation d'implémentation avec des variables continues.

Tableau 3.1 – Comparaison des méthodes d'apprentissage par renforcement.

3.3.3.2 Principaux algorithmes d'apprentissage par renforcement

a. Algorithme TD(0)

En utilisant la différence temporelle, on obtient directement l'algorithme qui évalue une politique fixée π : c'est l'algorithme TD(0). Ce dernier 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.

Dans le cas de TD(0), les mises à jour se font localement à chaque fois que l'agent réalise une transition dans son environnement, à partir d'une information se limitant à son état courant «s_t», l'état successeur «s_{t+1}» et la récompense « r_{t+1}» reçue dans cet état successeur.

Il faut noter, par contre, que comme TD(0) estime la fonction de valeur d'un problème, faute d'un modèle des transitions entre les états, l'agent est incapable d'en déduire quelle politique suivre, car il ne peut réaliser un pas de regard en avant pour déterminer quel est l'état suivant de plus grande valeur. Ce point explique que l'on préfère avoir recours aux algorithmes qui travaillent sur la fonction de qualité (état/action) que nous allons présenter dans ce qui suit.

b. Sarsa:

Sarsa est une méthode d'apprentissage par renforcement similaire à l'algorithme TD(0) qui évalue et améliore la stratégie utilisée pour prendre les décisions d'action . Il travaille cependant sur les qualités des couples (s_t,a_t). Son équation de mise à jour est la suivante :

$$Q_{k+1}(s_i,a_i)=Q_k(s_i,a_i)+\alpha(r_{i+1}+\gamma Q_k(s_{i+1},a_{i+1})-Q_k(s_i,a_i)) \qquad (3.14)$$

L'information nécessaire pour réaliser une telle mise à jour est donc le quintuple (s_i, a_i, r_{i+1}, s_{i+1}, a_{i+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 s_i l'aura conduit dans l'état s_{i+1}.

c. Q-learning :

Cet algorithme 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 serait l'action à réaliser au pas de temps suivant. Son équation de mise à jour est la suivante : $$Q_{k+1}(s_i,a_i)=Q_k(s_i,a_i)+\alpha(r_i+\gamma \max_{a\in A(S_{i+1})} Q_k(s_{i+1},a_i)-Q_k(s_i,a_i)) \qquad (3.15)$$

De ce fait, il apparaît que l'algorithme Sarsa effectue les mises à jour en fonction des actions choisies effectivement alors que l'algorithme du Q-learning effectue les mises à jour en fonction des actions optimales ce qui est à la fois plus simple et plus efficace.

d- TD(λ), Sarsa(λ), Q-learning(λ) :

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 ; Sutton et al. 1998]. Cependant, ces algorithmes présentent le défaut de ne mettre à jour qu'une valeur à pas de temps à savoir de l'état que l'agent est en train de visiter.

On dressera alors le diagramme suivant des différents algorithmes d'apprentissage dérivés de méthodes TD.

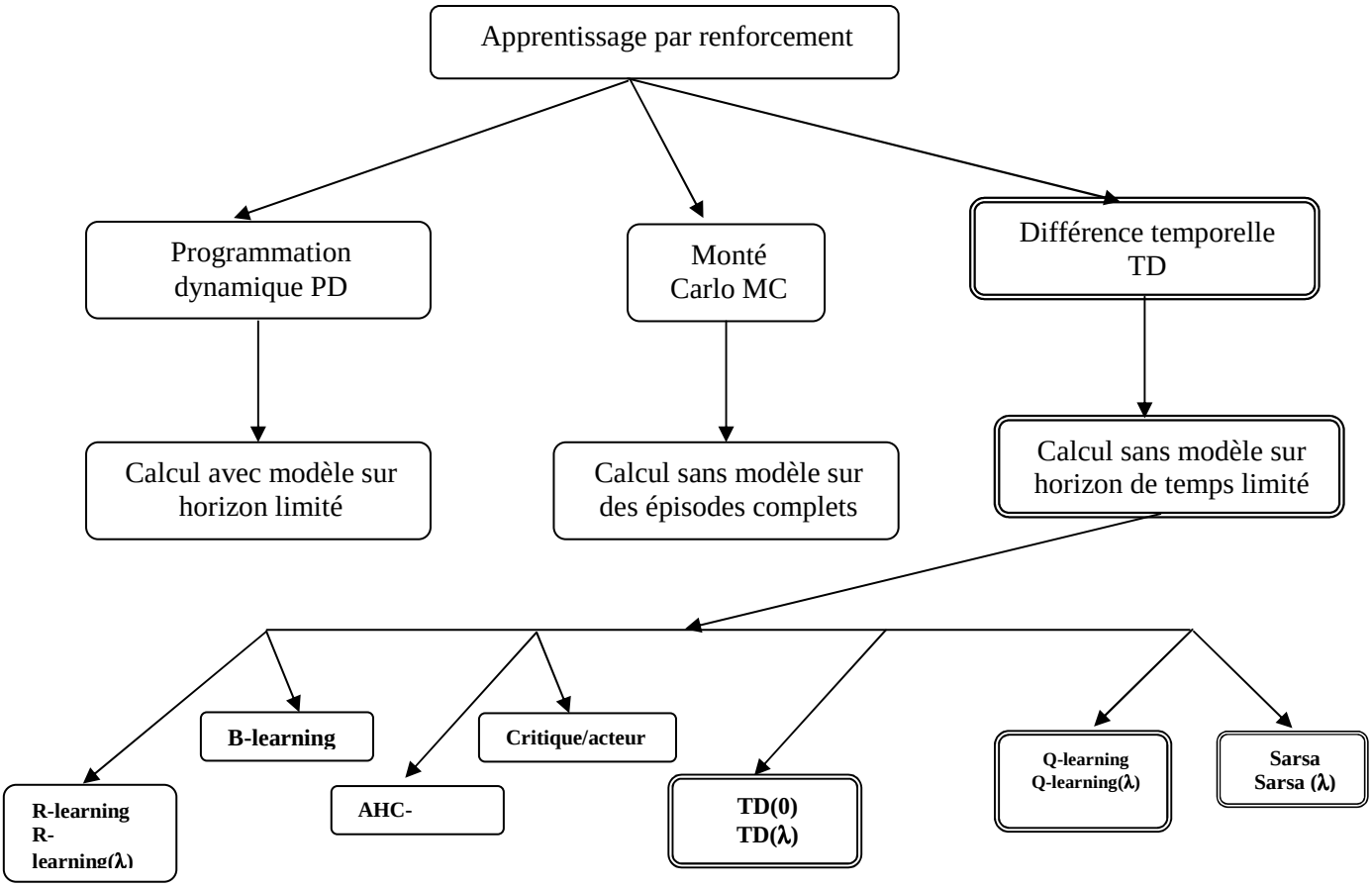


Figure 3.7 - Différents algorithmes d'apprentissage par renforcement

Discussion

L'apprentissage par renforcement se distingue de l'apprentissage supervisé par le fait que lorsqu'il reçoit le premier signal d'évaluation, l'apprenant ne sait toujours pas si la réponse qu'il a donné est la meilleure, il doit alors essayer d'autres réponses pour déterminer s'il peut recevoir une meilleure évaluation. Ceci induit une activité d'exploration de la part de l'agent. Il faut que ce dernier parcoure donc son environnement pour déterminer dans quelles circonstances il est puni ou récompensé et quelles sont les séquences d'actions qui lui permettent d'atteindre les récompenses plutôt que les punitions.

Cette nécessité d'explorer est à la source, dans tous les travaux de l'apprentissage par renforcement, du dilemme posé par le compromis entre exploration et exploitation.

3.3.4 Dilemme exploration - exploitation

Pour régler la politique de façon à maximiser sa récompense sur le long terme, la phase d'apprentissage se trouve confrontée à la nécessité de trouver un compromis entre l'exploitation qui consiste à refaire les actions dont on sait déjà qu'elles donnent lieu à une récompense et l'exploration qui consiste à parcourir de nouveaux couples (état, action) à la recherche d'une récompense cumulée plus grande mais au risque d'adopter parfois un comportement sous optimal. En effet, tant que l'agent n'a pas parcouru la totalité de son environnement, il n'est pas certain que la meilleure politique qu'il connaît soit la politique optimale.

En conséquence, toutes les preuves de convergence des algorithmes d'apprentissage par renforcement, exigent en théorie que chaque transition soit expérimentée un nombre infini de fois. En pratique et lorsque l'environnement est de grande taille, on se contente d'une exploration partielle qui suffit en général à découvrir une politique satisfaisante.

Cette présence d'un compromis «exploration - exploitation » induit donc l'existence d'un grand nombre d'algorithmes qui font des choix différents. La résolution d'un problème d'apprentissage par renforcement reste alors un art plutôt qu'une science [Jozefowiez, 2001].

Les méthodes d'exploration les plus employées sont d'une part les méthodes « Softmax » et d'autre part les méthodes « ε–greedy » ou « glouton » [Thurn, 1992; Kaelbling et al., 1996]

- ✓ Les méthodes Softmax : elles utilisent un facteur de température, noté T, au sein d'une distribution Boltzmann ou Gibbs. La probabilité de choisir une action a est égale à :

$$\frac{e^{\frac{Q(s,a)}{T}}}{\sum_{b \in A} e^{\frac{Q(s,b)}{T}}}$$

où T est un facteur (Température) pris assez élevé au début puis décroît vers 0 au fur et mesure que l'apprentissage progresse.

- ✓ Les méthodes ε–greedy : elles choisissent l'action optimale (appelée action « gourmande ») avec une probabilité 1-ε et tire une commande au hasard avec une probabilité ε.

Néanmoins, ces différentes méthodes d'exploration peuvent être utilisées indifféremment et quelque soit l'algorithme d'apprentissage par renforcement par différence temporelle auquel on fait appel.

3.4 Méthodes de généralisation

Les algorithmes d'apprentissage par renforcement utilisent des fonctions de valeur explicitées sous forme de tableaux (autant de lignes que de situations et autant de colonnes que d'actions, les cases de la matrice sont les fonctions de valeurs). Et chaque couple (état, action) doit être visité une infinité de fois pour assurer la convergence [Jaakkola, 1994]. Ceci est efficace pour des problèmes simples, mais dans les processus avec un très grand nombre d'états, ces tableaux peuvent prendre des tailles énormes : on tombe dans la malédiction de la dimensionnalité [Bellman, 1957].

Du fait que l'espace des situations possibles combiné avec celui des actions potentielles est si grand, une exploration exhaustive de toutes les paires de situation -action peut s'avérer impossible (nécessite un temps énorme pour la convergence) de même qu'une mémorisation exhaustive. Il est alors nécessaire d'utiliser des techniques de généralisation ou d'approximation de fonction qui permettent d'attribuer une valeur à des états qui n'ont pas été rencontrés précédemment.

Plusieurs techniques d'approximation ont été utilisées. L'idée de base est de remplacer le tableau qui stocke les fonctions de valeurs par une fonction définie par un nombre de paramètres inférieurs au nombre d'états.
Et pour approximer cette fonction, il suffit d'utiliser des exemples de la véritable fonction de valeur que l'on cherche à approcher et en déduire une fonction entière or, on ne connaît pas la véritable fonction de valeur.

A Chaque mise à jour de cette fonction par les méthodes d'apprentissage par renforcement correspond à une modification de la valeur de l'état courant vers une valeur qui est plus représentative. Ces mises à jour de valeur serviront comme exemple d'entraînement de méthode d'apprentissage supervisé pour obtenir un approximateur de fonction de la fonction recherchée. Dans cette optique, il existe plusieurs approches différentes [Sutton et al. 1998 ; Desjardins, 2007], toutefois, l'approche la plus employée est la méthode basée sur le gradient non linéaire.

En effet, le réseau de neurones est capable d'approcher, avec précision, n'importe quelle fonction régulière. Avec sa structure compacte, il permet d'obtenir un excellent apprentissage sur des espaces à très grandes dimensions et avec peu de paramètres [Touzet , 1990 ; Lin, 1992 ; Sehad et al., 1995, Touzet, 1998 ; Benhamza et al., 2007a; Benhamza et al., 2007c]

Une étude détaillée des différentes méthodes d'approximation se trouve dans [Albus, 1975 ; Bertsekas et al, 1996 ; Kaelbling et al., 1996 ; Munos, 1997 ; Sutton et al. 1998 ; Coulom, 2002]

3.5 Paramètre d'apprentissage:

On a vu précédemment que la formule de mise à jour de la fonction de valeur V et Q (équations 3.12 et 3.13) dépend de plusieurs facteurs qui jouent 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.
Nous allons détailler ces paramètres pour en déduire l'intérêt de chacun :

- ✓ Le paramètre γ ($0 \leq \gamma < 1$) 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 de 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 suivent.

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 ($0<\alpha\leq 1$) est un coefficient dénommé « taux d'apprentissage » et 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 dans l'équation (3.12), où «n» est le nombre de visites de «s» depuis le début [Bertsekas et al, 1996].

(2)- on prend aussi $\alpha = 1/t$ donc variable au cours du temps, initialement grand et tendant vers zéro [Bertsekas et al, 1996].

(3)- Pour beaucoup de problèmes, le choix optimal de α est 1, selon [Kaelbling et al., 1996] ceci permet d'accélérer l'apprentissage.

(4)- Dans [Dayan, 1992; Dayan et al., 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. Mataric [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. D'autres méthodes utilisent les estimateurs de progrès qui consistent à incorporer des conseils directement dans la fonction de renforcement. 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 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 bien des manières différentes. Si on adopte la méthode « ε-greedy ». 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 pour avoir trop d'action aléatoires pour avoir un compromis entre apprentissage et exécution de la tache.

Discussion

Finalement, nous constatons qu'aucune de ces études ne présente une initialisation générique de ces valeurs car en pratique, on choisit les fonctions de renforcement d'une façon intuitive, les valeurs initiales des paramètres d'apprentissage d'une façon arbitraire et de la table Q ou V, on l'initialise le plus souvent à zéro.

Avant de terminer ce chapitre, il serait utile de présenter le domaine d'application de l'apprentissage par renforcement.

3.6 Les domaines d'application de l'apprentissage par renforcement

L'apprentissage par renforcement est devenu, en quelques années, une des disciplines la plus active de la communauté de l'intelligence artificielle. A l'interface entre apprentissage, automatique, sciences cognitives, l'apprentissage par renforcement utilise des techniques variées pour aborder le problème de l'acquisition d'un comportement optimal dans un environnement incertain et dynamique.

Les méthodes de l'apprentissage par renforcement présentées ici sont adaptées pour les problèmes de décisions séquentielles avec des récompenses immédiates ou retardées. Dans ce cadre là, de nombreuses applications de l'apprentissage par renforcement ont été développées.

Les jeux ont bien sur fait l'objet des premières applications en intelligence artificielle du principe de renforcement. L'exemple le plus classique est le programme de dames américaines « Checkers » d'Arthur Samuel [Samuel, 1959]. D'autres exemples ont suivi comme le programme de BackGammon « TD-Gamon » de Tesauro [Tesauro, 1994] qui a atteint un niveau de classe mondiale en intégrant des techniques d'apprentissage par renforcement. D'autres problèmes de jeux plus complexes sont abordés tels que les échecs et le Go [Baxter, 1998].

Plus récemment, les applications industrielles portant sur l'optimisation de la conduite de systèmes de production, de communication sont de plus en plus nombreuses telles le routages des paquets [Boyan, 1993], l'ordonnancement des taches [Zhang et al, 1995], la gestions des applications réparties [Girard-Faugère, 1997], la conduite de cultures agricoles [Ndiaye, 1999] ainsi que l'optimisation de la maintenance d'une constellation de satellites [Cabarbaye et al., 1999].

L'autre grand classique des applications de l'apprentissage par renforcement est la robotique. Citons quelques exemples : les robot jongleurs, balanciers [Schaal, 1994; Kimura, 1997], l'apprentissage de tâches d'assemblage [Asada et al, 1991]. Passant par son utilisation pour la gestion du dialogue sur un robot parlé, ils ont été également utilisés afin de résoudre des problèmes de navigation [Simmons et al., 1995 ; Laroche, 2000]. En robotique exploratrice, nous pouvons citer les travaux de Bernstein [Bernstein et al., 2001; Cardon et al., 2001] qui ont montré l'adéquation de ce formalisme pour la résolution des problèmes décisionnels auxquels est confronté un robot autonome évoluant sur Mars.

Modélisation

Conclusion

L'objectif de cette partie était de présenter les bases théoriques générales de l'apprentissage par renforcement. En guise d'introduction, nous avons commencé par replacer l'apprentissage par renforcement dans le contexte qui lui a donné naissance. Nous avons présenté ensuite le cadre général des modèles de Markov, qui permettent de formaliser les phénomènes observés empiriquement et de modéliser l'interaction de l'agent avec l'environnement.

Une fois ce modèle posé, nous avons exposé les principaux algorithmes de planification et d'apprentissage qui s'appliquent dans ce cadre en étudiant leurs avantages et leurs inconvénients à la lumière de notre problématique. Parmi le grand nombre d'algorithmes issus des méthodes Différence temporelle (TD), nous avons décrit brièvement les plus connus et nous avons insisté sur les algorithmes les plus utilisés.

Enfin, Nous pourrons dire que ces algorithmes élémentaires sont clairement limités à des systèmes dont l'espace d'états est de faible dimension mais en utilisant d'autres structures comme les réseaux de neurones artificiels, les fonctions polynomiales ou d'autres outils statistiques, on pourrait surpasser ce problème pour finalement permettre la généralisation de l'apprentissage à de systèmes complexes à grand nombre d'états.

Nous avons traité aussi le problème du dilemme Exploration- Exploitation particulièrement sensible pour les systèmes de grandes dimensions et avons cité les méthodes d'exploration telles « softmax » et «ε- greedy »

Un point important en apprentissage par renforcement est le choix des paramètres d'apprentissages. En effet, bien que les propriétés de convergence des ses algorithmes ont été largement étudiées, peu de règles précises existent pour choisir correctement ces derniers et peu de travaux ont été élaboré pour l'estimation de la fonction de renforcement et des valeurs initiales de la table d'utilité Q.

En effet, c'est un problème qui reste toujours ouvert et pour lequel nous allons essayer de donner notre contribution dans la partie mise en œuvre suivante.

Nous avons terminé cette partie par l'exposition des différents domaines d'application de l'apprentissage par renforcement.

Partie C

Mise en oeuvre

Introduction

Dans cette partie, comportant le chapitre IV et V, nous allons chercher à élaborer un processus automatique qui permet de créer une entité pouvant prendre, individuellement des décisions, de façon à obtenir un comportement rationnel et autonome. L'outil utilisé est l'apprentissage par renforcement (AR), l'algorithme implémenté est le Q-learning et l'application choisie est la robotique mobile.

Nous développerons ensuite une étude formelle appuyée par des tests de simulation qui permettra de proposer des justifications expérimentales pour le choix, d'une part de la fonction de renforcement et d'autre part des valeurs initiales des paramètres d'apprentissage et de la table Q. Les effets de ces paramètres sur la stratégie sont discutés afin de suggérer une analyse générique. Nous essayerons de découvrir l'influence de ces éléments sur le processus d'apprentissage.

Partie C

Mise en oeuvre

Chapitre IV

Conception

Chapitre IV

Conception

4.1 Introduction

Dans la partie précédente, nous avons traité une part de la problématique de notre travail : la modélisation de la conception d'agent intelligent avec apprentissage par renforcement. Dans ce chapitre, nous allons exposer la structure de notre modèle de conception d'agent tout en définissant le principe de fonctionnement de chaque module intégré pour spécifier ensuite le choix des valeurs initiales des paramètres d'apprentissage.

4.2 Modèle général des algorithmes d'apprentissage par renforcement

À l'inverse de la programmation dynamique, les algorithmes d'apprentissage par renforcement sont exécutés en temps réel. Plusieurs algorithmes ont été décrits dans la partie B. Ils se basent sur une comparaison entre récompense que l'on s'attend à recevoir et la récompense que l'on reçoit effectivement selon l'équation (3.12).

Ces approches se distinguent par le type de fonction valeur estimée et par les techniques d'évaluation et d'amélioration des stratégies [Sutton et *al.*, 1998] néanmoins leur modèle général est semblable et schématisé par la figure suivante :

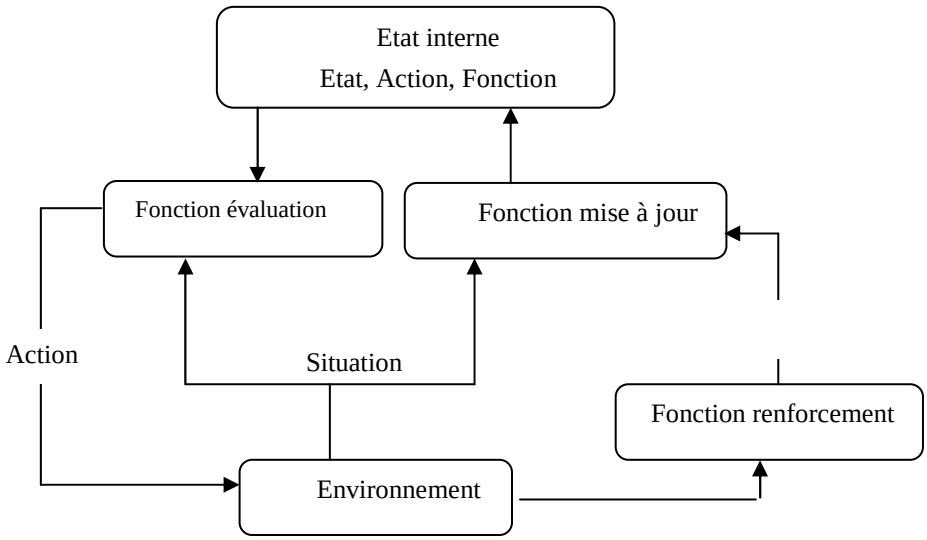


Figure 4.1- Modèle général pour les algorithmes d'apprentissage par renforcement

Une étude détaillée des différents algorithmes existe dans la littérature scientifique [Watkins et *al.*, 1992 ; Kaelbling et *al*., 1996; Sutton et *al*. 1998]. Un tableau descriptif a été dressé et présenté en annexe I.

4.3 Choix de l'algorithme

Selon l'étude sus référenciée, notre choix s'est orienté vers l'algorithme Q-learning. Il a été introduit par Christopher Watkins en 1989 et il est, sans doute, l'algorithme le plus usuel dans la communauté de l'apprentissage par renforcement.

Cette caractéristique vient, en raison de la simplicité, l'efficacité et les preuves formelles de convergence qui ont accompagné la publication de cet algorithme [Watkins, 1989 ; Watkins et al, 1992 ; Sutton et al. 1998].

En effet, le Q-learning est très simple [Jaakkola et *al.*, 1994] : la stratégie optimale est apprise de manière implicite sous la forme d'une fonction de valeur Q.

Additionnellement, les mises à jour de la fonction de renforcement sont effectuées en fonction des actions optimales, la solution est ainsi atteinte plus rapidement par rapport aux autres algorithmes ce qui rend l'algorithme plus efficace.

Et enfin, sa convergence a été prouvée dans le cas des processus de décision markoviens [Bertsekas, 1989 ; Sutton et *al.*, 1998 ; Watkins et *al.*, 1992] et [Jaakkola, 1994]. Selon ces auteurs, cet algorithme converge presque sûrement vers la fonction valeur pourvu que la trajectoire (s_t, a_t) soit récurrente dans SxA, que S et A soient finis et que (α_t) satisfasse aux hypothèses ordinaires de l'approximation stochastique. Ces conditions se résume comme suit:

Pour que le Q-learning converge, il faut que :

- ✓ Les ensembles S (états) et A(actions) soient finis ;
- ✓ Chaque état soit visité une infinité de fois ;
- ✓ $\sum_{t}^{\infty} \alpha_t(s) = +\infty$ et $\sum_{t}^{\infty} \alpha^2{}_t(s) < +\infty \quad \forall s \in S.$

Principe

Nous avons validé notre étude effectivement avec l'algorithme Q-learning dans lequel, une fonction de valeur $Q^{\pi}(s,a)$ est estimée et mémorisée durant le processus d'apprentissage. La fonction de valeur représente la somme des récompenses futures espérées que l'agent souhaite recevoir en exécutant l'action «a» depuis l'état «s» et en suivant la stratégie «π». C'est la stratégie optimale recherchée par l'agent.

Cet algorithme est fondé sur une approche itérative de résolutions dont la fonction de valeur de l'action optimale Q* est l'unique solution de l'équation de Bellman (3.10). Son équation de mise à jour est :

$$Q(s,a) := Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Où «r» est la récompense reçue pour la transition de l'état « s » au nouvel état « s' » après l'exécution de l'action « a ».

4.4 Notre modèle de conception

Le principe de notre modèle de conception d'agent intelligent par Q-learning est fondé sur le schéma suivant [Benhamza et *al.*, 2007b ; Benhamza et *al.*, 2007c]:

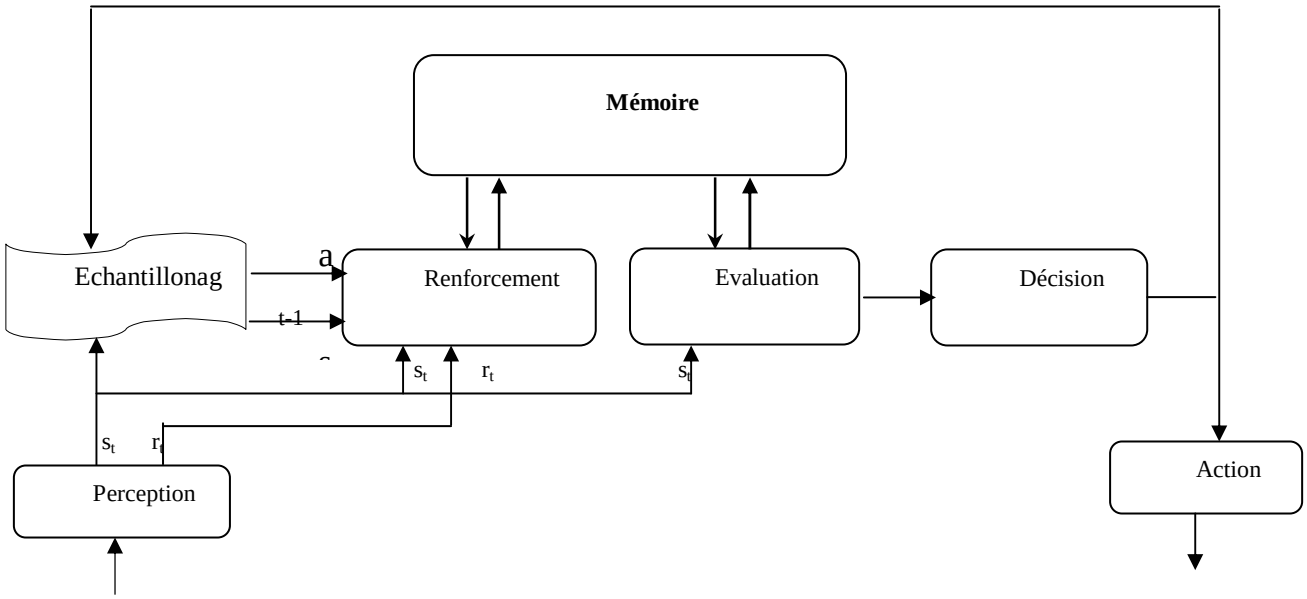


Figure 4.2 - Principe de conception d'agent par Q-learning

Il est constitué des quatre modules principaux : le module de mémorisation, le module de renforcement, le module d'évaluation et le module de décision.

Le fonctionnement de ce modèle peut être synthétisé comme suit : dans le module mémoire, on associe une estimation de l'espérance de gain pour chaque couple (situation, action). Le module de renforcement procède à un apprentissage par renforcement de type Q-learning et met à jour le contenu de la mémoire avec ses informations disponibles au temps « t ».

La fonction d'évaluation extrait de la mémoire une estimation de l'utilité de chaque action dans l'état actuel et enfin, l'action optimale est déterminée dans le module décision selon le critère d'exploitation et exploration choisi.

Nous allons maintenant exposer en détail, chacun de ces modules.

4.5 Description des différents modules

1- Module mémoire

Dans ce module, la mémoire permet de stocker la valeur d'utilité courante. Nous utilisons implicitement un tableau (statique ou dynamique) de dimension : nombre d'états (S) *x nombre d'actions* (A). Ce dernier enregistre donc la valeur d'utilité Q(s,a) pour toutes les situations possibles de S et toutes les actions de A.

Cette fonction de valeur Q, qui à chaque couple (situation, action), associe une valeur réelle telle que : Q : Etat x action $\rightarrow$ R

L'emploi de cette structure est nécessaire pour établir la convergence du Q-Learning.

2- Module de renforcement

La fonction de renforcement du Q-Learning est fondée sur l'algorithme d'itérations sur les valeurs dont l'objectif est d'optimiser la fonction d'utilité pour chaque couple *état-action (s,a)* afin d'en déduire une stratégie de commande optimale.

Le rôle de ce module est d'optimiser donc l'estimation de gain Q(s,a) pour une situation « s » et une action « a ». Pour ce faire, nous utilisons l'équation de mise à jour du Q-learning suivante : $Q_{k+1}(s_i, a_i) = Q_k(s_i, a_i) + \alpha(r_i + \gamma \max_{a \in A(S_{i+1})} Q_k(s_{i+1}, a_i) - Q_k(s_i, a_i))$

3 - Module d'évaluation

Le but de ce module d'évaluation ou d'estimation est de faire le lien entre la mémoire et la fonction de décision. Il s'agit simplement de fournir à la fonction de décision, l'estimation courante de l'espérance de gain de chaque action pour l'état actuel du système.

4- Module de décision

Nous avons vu que la convergence vers l'optimum de l'algorithme Q-learning était assurée sous la seule contrainte de parcourir indéfiniment l'ensemble des états et actions possibles (Table d'utilité Q).

A chaque instant, l'agent doit choisir la commande à effectuer. Il est donc nécessaire de réaliser un compromis entre exploitation et exploration. Deux méthodes ont été précédemment citées. Dans notre cas, nous avons opté pour une fonction de décision de type ε- glouton. Celle-ci retient le principe qui consiste à choisir l'action gourmande la plupart du temps, tout en choisissant plus ou moins régulièrement une commande aléatoire.

Discussion

L'architecture de notre modèle de conception d'agent intelligent étant proposée il faudrait ensuite spécifier les valeurs d'initialisation des paramètres dans le module d'apprentissage pour une finalisation correcte de cette dernière.

A cet effet, le choix d'une application est obligatoire et doit être fait dés le départ pour pouvoir cerner le comportement de l'agent et faire ressortir ainsi l'influence et l'impact de ces paramètres sur la stratégie optée.

4.6 Choix de l'application : Robotique mobile

L'applicabilité de notre modèle de conception est choisie sur une plate-forme de robotique mobile autonome qui permettra d'une certaine manière de présenter sa certification. Le programme expérimental, de même que les tests de simulation serviront ensuite d'appui pour le choix des valeurs initiales et seront présentés dans le chapitre suivant.

4.6.1 Définition du cadre expérimental

On expérimentera ainsi sur une application simulée de robotique mobile consacrée à la navigation d'un robot dans un labyrinthe. Le problème posé dans ce type d'application est que le robot ou l'agent doit évoluer dans un environnement inconnu en évitant les obstacles qu'il peut rencontrer pour atteindre la cible.

Il ne s'agit pas de réaliser une carte du plus court chemin mais surtout de définir un comportement rationnel et intelligent permettant au robot de se déplacer en toute sécurité dans son environnement. Nous recherchons ainsi, une estimation autonome de sa performance, et une capacité d'autoévaluation de cette dernière.

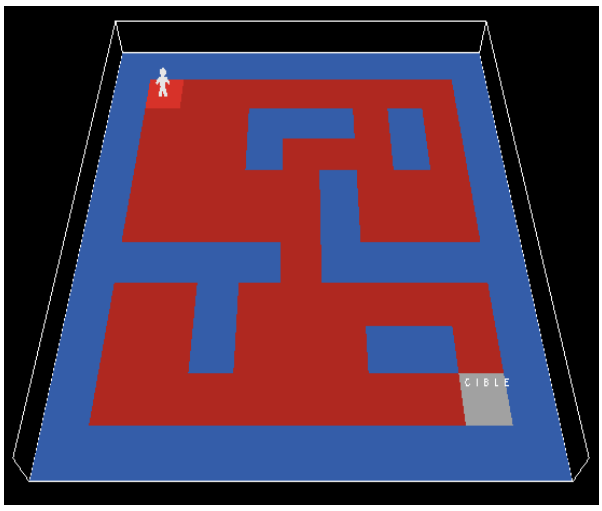


Figure 4.3 - Robot Navigateur

a. Labyrinthe de test

On a choisi un labyrinthe de 121 cases (11x11) avec et sans murs qui sont créés d'une façon aléatoire (Figures 4.4.a et 4.4.b).

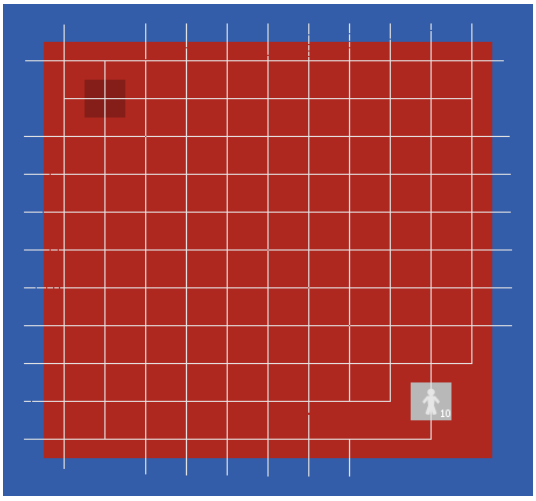


Figure 4.4.a- Labyrinthe
de configuration initiale sans mur

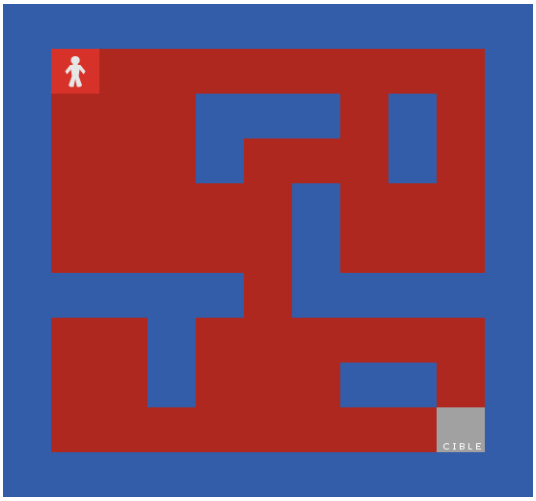


Figure 4.4.b- Labyrinthe
de configuration initiale avec mur

b. Description de la tache

L'environnement est un rectangle découpé en un ensemble régulier de cases, chacune associée à un état du système. Chaque case est soit vide, soit occupée par un mur ou par la cible.

L'agent possède quatre déplacements (actions) possibles: aller dans la case adjacente de droite, de gauche, en haut ou en bas. De plus, on suppose qu'il connaît exactement l'état dans lequel il se trouve à tout moment. Mais, il ne sait pas à priori où sont les obstacles et la cible (les bords du labyrinthe sont considérés comme des obstacles, dans lesquels le système peut se cogner).

L'objectif est d'apprendre une stratégie pour atteindre la cible présente dans le ''labyrinthe'' en utilisant les quatre actions de base

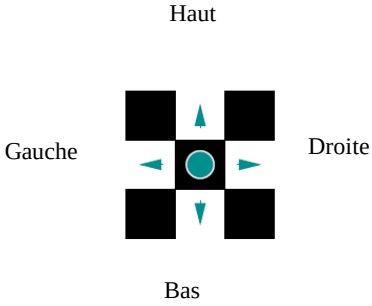


Figure 4.5- Les actions possibles

La position de départ de l'agent et la cible ont été configurées dés le début Si l'agent atteint l'objectif, l'épisode est terminé et l'agent est replacé sur sa case initiale. Le chemin le plus court entre l'état initial et l'état cible mesure 16 cases. La fonction Q est initialisée à une valeur réelle Q_{init}. Lorsqu'un agent rencontre un mur, il se cogne et il est replacé dans sa case courante.

L'objectif visé est de joindre effectivement, à chaque état l'action la mieux appropriée et d'associer ainsi à l'agent un comportement qualifié par l'extérieur comme intelligent.

4.6.2 Comportement désiré de l'agent

Dans le cas de l'application choisie, l'agent est mobile et doit atteindre une cible ou se déplacer au hasard si la cible n'est pas connue, tout en évitant les obstacles qu'il rencontre. Ce comportement « intelligent » peut être vu comme la coordination de ces trois comportements :

- ✓ atteindre un but
- ✓ se déplacer au hasard
- ✓ et éviter les obstacles

Chacun de ces comportements se traduit par une sélection d'action selon la récompense attribuée à chacun de ces derniers, autrement dit en fonction du gain reçu.

On propose d'appréhender le problème de sélection d'actions d'un agent en fonction de l'état du monde (l'agent et son environnement) pour choisir la séquence d'actions les plus adaptées à l'accomplissement des buts de l'agent (Figure 4.6)

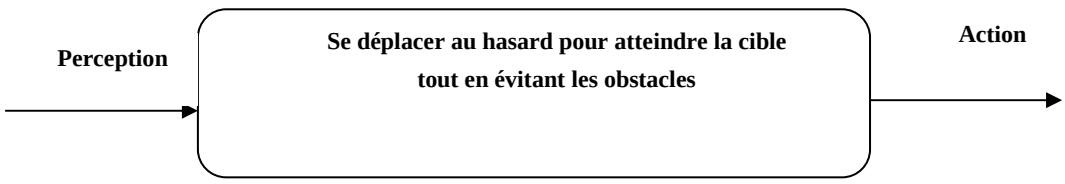


Figure 4.6 - Comportement global de l'agent

Ce comportement global peut se traduire par décomposition en plusieurs comportements de base :

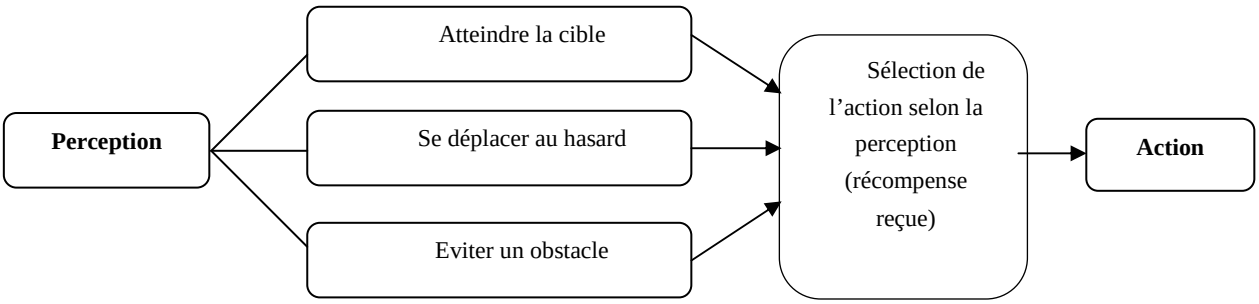


Figure 4.7 - Notre approche comportementale du modèle basé Q-learning

Un comportement complexe peut être donc simplifié par le principe de la sélection d'action (Figure 4.7). Il faut seulement que l'agent puisse choisir son comportement en fonction de sa conséquence. Pour cette raison, l'algorithme du Q-learning est une bonne implémentation de la sélection du comportement.

Cette architecture comportementale permet de démontrer qu'il est possible aussi de produire un comportement intelligent par coordination de plusieurs comportements élémentaires en prenant pour base un agent réactif simple et autonome sans recourir à la planification ou à la programmation.

La figure 4.8 présente l'algorithme résumant le fonctionnement du modèle qui est constitué des instructions suivantes :

Début

Initialiser Q(s,a) arbitrairement ;

Choisir le point de départ initial;

Pour chaque épisode

Tant que «s» n'est pas l'état but faire

Choisir « a » selon la stratégie d'exploration ε- glouton et en fonction de Q(s,a)

a := décision (une des actions: haut/droit/bas/gauche) selon la stratégie optée (optimale ou aléatoire)

Exécuter «a» et observer le nouvel état «s» et la récompense «r »

Mettre à jour la table utilité Q selon :

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a))$$

s :=s'

Fin tant que

Fin

Figure 4.8 - Algorithme de résolution basé Q-learning pour robot navigateur

La sélection de l'action se fait selon un critère d'exploration/exploitation. Nous avons utilisé la méthode ε-glouton dans laquelle la probabilité de choisir une action aléatoire est ε, sinon, l'action sélectionnée est celle ayant, pour l'état courant, la plus grande valeur de Q.

Cette fonction de décision ε- glouton permet de maîtriser le taux d'exploration et nous remarquons que la stratégie converge vers des solutions optimales.

L'algorithme de la fonction de décision ε- glouton est présenté ci-dessous :

Entrée : Q_t
Sortie :a_t
Début
Choisir un nombre aléatoire «ξ» tel que :
ξ : = nombre réel aléatoire ∈ [0,1[
Si ξ < ε
alors a_t := une action aléatoire (haut/droit/bas/gauche)
Sinon a_t := argmax $_{v \in actions}$ *Q(s,v) ; (action optimale)*
Fin si
Fin

Figure 4.9 - Algorithme de la fonction de décision (ε- glouton)

4.7 Conclusion

Dans ce chapitre, nous avons établi les principes de notre modèle de conception d'agent intelligent par Q-learning ainsi que les différents modules intégrés. Nous avons défini aussi le comportement de l'agent navigateur dans un labyrinthe basé sur le principe de la sélection d'action. Nous avons montré que les architectures comportementales permettent de réaliser des taches complexes à l'aide de modules élémentaires plus simple à concevoir.

L'objectif du chapitre suivant est de valider notre modèle de conception pour différentes valeurs des paramètres d'apprentissage.

Chapitre V

Expérimentations & Résultats

Chapitre V
Expérimentations & résultats

5.1 Introduction

Dans le chapitre 4, nous avons présenté le principe de notre modèle de conception d'agent intelligent et les différents modules intervenants. Ce chapitre 5 est consacré à l'étude comportementale de cet agent sur l'exemple de labyrinthe dans le cadre de la robotique mobile.

Ainsi, nous proposons d'évaluer la performance de cet algorithme en réalisant des tests de simulations. Les résultats fournis vont nous permettre de valider ce principe de conception et de mieux appréhender les avantages et les limites de notre approche d'initialisation dans la stratégie optée par l'agent [Benhamza et *al*., 2007b, Benhamza et *al.*, 2007d].

5.2 Présentation du problème

L'évaluation de l'approche de conception et d'initialisation ainsi que la simulation de la stratégie a été testée avec le logiciel NetLogo : logiciel permettant de faire usage de la simulation basée agents pour les phénomènes dynamiques et complexes. Une description concise de cet environnement est présentée en Annexe II.

Avant de commencer, il s'avère nécessaire de donner quelques détails complémentaires sur les paramètres utilisés.

1. Nombre d'itérations ou d'épisode

Un épisode correspond à une simulation entre l'instant où le système se trouve dans l'état initial et l'instant où le système se trouve dans l'état but. Ce dernier a été réglé par défaut à la valeur 1000. Cette valeur donne de bons résultats, quelle que soit la taille de la carte sélectionnée (10, 15 ou 20). Pour une carte présentant une situation plus complexe, il peut s'avérer nécessaire d'augmenter cette valeur.

2. Nombre de pas

C’est le nombre de périodes d'échantillonnage entre l'instant où le système se trouve dans l'état initial et l'instant où le système se trouve dans l'état but.

3. Taux d’exploration ε

Pour notre application, on définit un "taux d'exploration" égal à 0,1 et au moment où l'agent doit choisir une action à accomplir, il tire au hasard un nombre réel compris entre 0 et 1. Si le nombre tiré est inférieur au taux choisi, la prochaine action sera choisie aléatoirement par l'agent (il tire aléatoirement parmi toutes les actions disponibles pour l'état dans lequel il se trouve), sinon, il choisit comme action, celle considérée comme étant la meilleure pour l'état dans lequel il se trouve.

4. Le taux d'apprentissage α

Il symbolise la capacité de mémorisation instantanée de l'agent. Normalement, ce dernier doit décroître au fil des itérations, pour atteindre finalement la valeur zéro (0.0) en fin d'apprentissage. Mais en pratique, on utilise souvent un coefficient fixe, cela permet un apprentissage permanent. Nous discuterons de ce choix dans la suite de ce chapitre.

5. Le taux d’actualisation γ

Il s'agit d'un paramètre intrinsèque au Q-Learning. 0.9 étant une valeur par défaut la plus utilisée dans les applications. Nous discuterons de la valeur initiale de ce paramètre d’apprentissage.

6. Renforcement r : Récompense ou pénalité

On général, on affecte une valeur positive à «r» pour une récompense, négative pour un échec et éventuellement nulle pour les autres cas. Ces valeurs ont une grande influence sur les résultats de l’apprentissage. Nous risquons d'être confrontés avec la difficulté de fixer au mieux la valeur des renforcements si nous nous intéressons à l'utilisation de notre modèle dans le cadre de systèmes complexes. C'est un choix largement exclusif.

☞ Objectif :

Dans le cadre de la robotique mobile, navigation dans un labyrinthe, nous proposons de développer des tests de simulation basés sur une étude formelle pour l’étude du comportement d’un agent conçu par Q-learning et discuter ainsi de la stratégie optimale optée pour différentes initialisations de paramètres d’apprentissage. Cette étude comportementale est représentée par un agent évoluant dans un labyrinthe de 121cases (11 x 11) avec et sans mur. Ces derniers sont crées aléatoirement.

5.3 Etude comportementale de l'agent conçu par Q-learning

5.3.1 Cas du labyrinthe sans obstacles (sans mur)

Dans ce cas, seules les transitions qui mènent à l'état but donnent une récompense non nulle.

On pose alors pour tout état « s » et action « a »:

$$R(s,a,s') = \begin{cases} r & \text{si } s' = \text{état but} \\ 0 & \text{sinon} \end{cases}$$

5.3.1.1 Influence de la valeur de fonction de renforcement « r »

5.3.1.1.1 Comportement avec récompense positive (r > 0) et 0<γ<1

Nous avons fait plusieurs tests avec plusieurs valeurs positives pour le renforcement « r » et une variation du facteur γ sur l'intervalle]0,1[, on remarque que la stratégie optimale tend à rapprocher l'agent de l'état but (figures 5.1.a et 5.I.b).

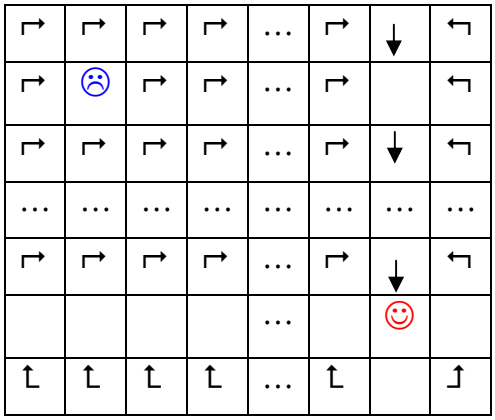


Figure 5.1.a - Stratégie de l'agent avec r > 0 et 0<γ<1

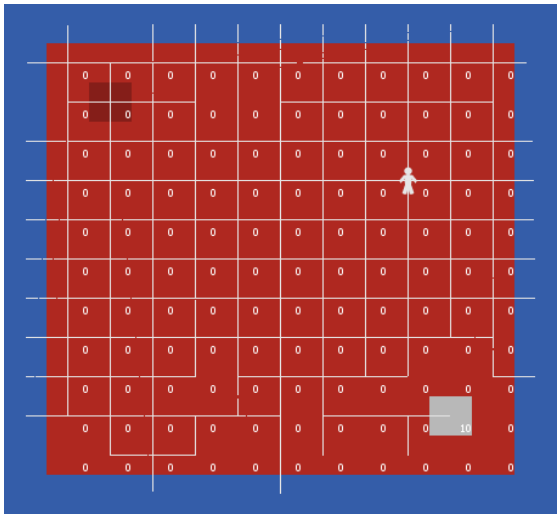


Figure 5.1.b-Simulation de la stratégie de l'agent avec r>0 et 0<γ<1

La figure 5.2.a donne la courbe d'apprentissage et montre l'évolution du nombre de pas par épisode en fonction du nombre d'épisodes. Après environ 30 épisodes, la trajectoire de l'agent est optimale (figure 5.2.b).

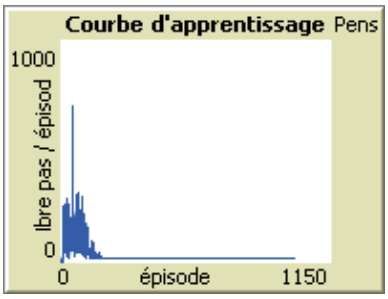


Figure 5.2.a - Courbe d'apprentissage sur l'exemple du labyrinthe
Evolution du nombre de pas par épisode en fonction du nombre d'épisode pour **r =10 γ= 0,9**

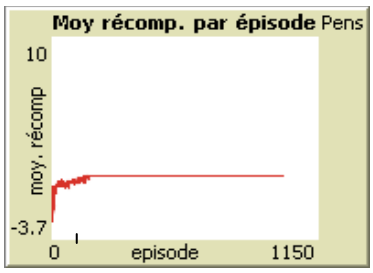


Figure 5.2.b - Moyenne de récompense par épisode sur l'exemple du labyrinthe

Donc avec un renforcement positif et un facteur d'actualisation compris dans l'intervalle]0,1[, l'apprentissage converge rapidement vers le trajet le plus court entre l'état initial et l'état but. Les actions fournies par l'algorithme conduisent à des trajectoires égale à 16 pas (figure 5.3).

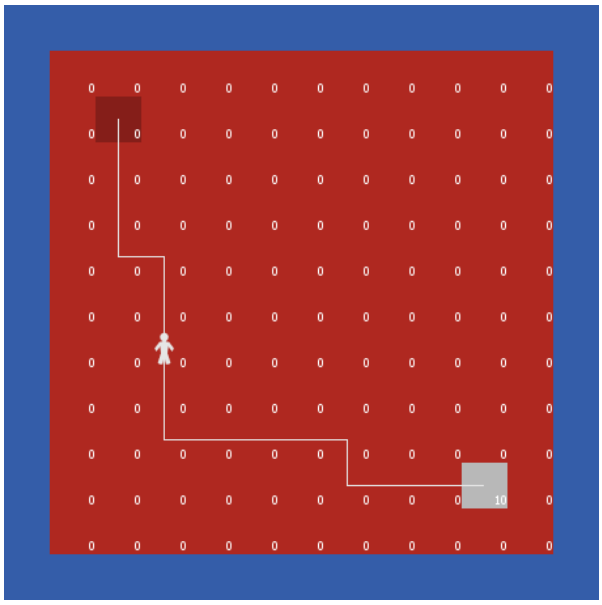


Figure 5.3 - Stratégie du plus court chemin.

5.3.1.1.2 Comportement avec récompense positive et un facteur d'actualisation nul

(r > 0 et γ = 0)

Si on prend dans ce cas, un facteur d'actualisation nul, la stratégie optimale contraint uniquement les actions des états voisins de l'état but (Figure 5.4.a).

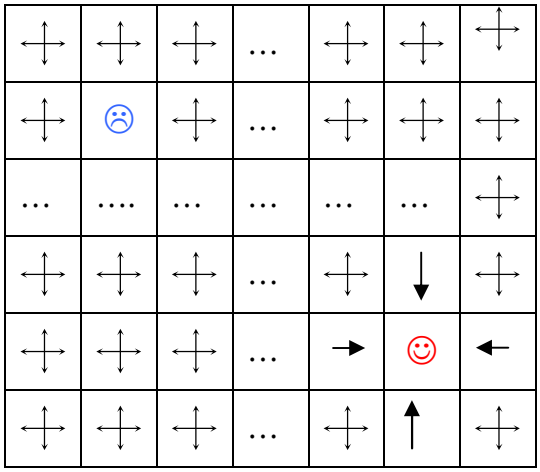


Figure 5.4.a- Stratégie de l'agent avec **r > 0 et γ=0** (Effet local)

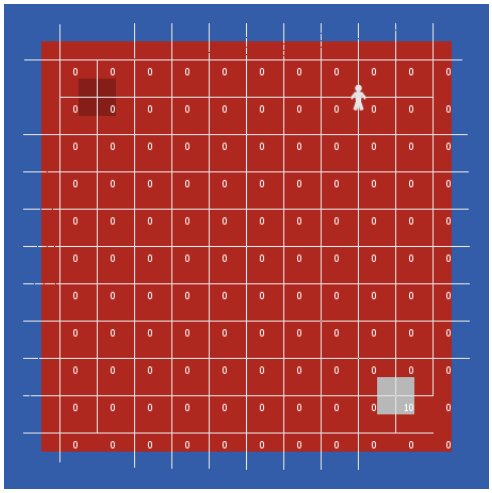


Figure 5.4.b - Simulation de la stratégie de l'agent **avec r > 0 et γ=0**

Ce comportement permet d'imposer une contrainte locale, en effet :

- ✓ Si le système passe par un état voisin de l'état but, il est attiré.
- ✓ Dans les autres états, il est libre de toute influence (Figure 5.4.b).

Pour ce labyrinthe (11x11), on remarque qu'après environ 70 épisodes, l'agent atteint la cible mais à chaque fois avec des nombres de pas différents Figure 5.5).

La courbe d'apprentissage (Figure 5.5.a) montre que l'apprentissage est tout à fait aléatoire.

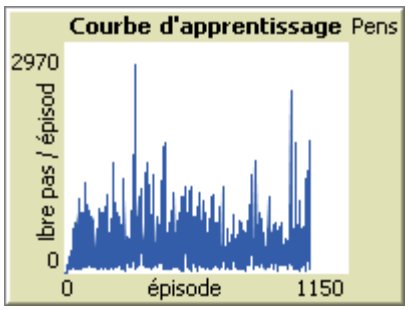


Figure 5.5 - Courbe d'apprentissage sur l'exemple du labyrinthe
Evolution du nombre de pas par épisode en fonction du nombre d'épisode pour **r = 10 γ= 0**

A cet effet et si on évalue l'utilité optimale Q(s,a) d'un couple état action (s,a) solution de l'équation de Bellman (3.10) pour $\gamma = 0$, celle-ci s'écrit alors :

$$Q^*(s,a) = \sum_{s' \in S} T(s,a,s')R(s,a,s')$$

et ne dépend donc que de la première récompense et pas des récompenses futures.

5.3.1.1.3 Comportement avec renforcement négatif (r<0)

On initialise le renforcement « r » à une valeur négative le plus souvent lorsqu'on désigne une situation critique ou un échec. Dans ce cas, quelle que soit la valeur de **γ,** la stratégie optimale est la même. L'agent évite les transitions associées à la récompense négative (figures 5.6.a et 5.6.b).

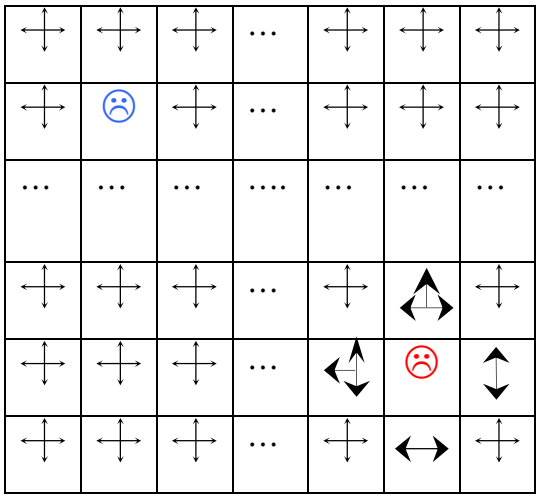


Figure 5.6.a - Stratégie de l'agent avec **r < 0**

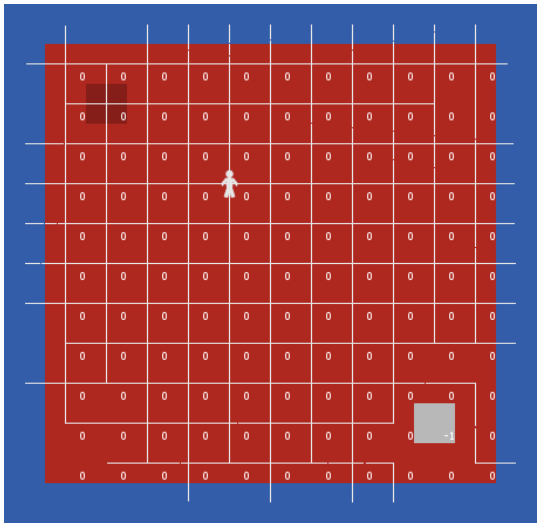


Figure 5.6.b - Simulation de la stratégie de l'agent avec **r<0**

Nous remarquons alors que :

- ✓ L'agent évite localement les transitions vers la cible (Figure 5.6.a et Figure 5.6.b).Ce comportement est antagoniste au cas (r > 0 et γ = 0).
- ✓ L'apprentissage n'est pas stable (Figure 5.7).
- ✓ Ce comportement avec récompense négative est donc à éviter.

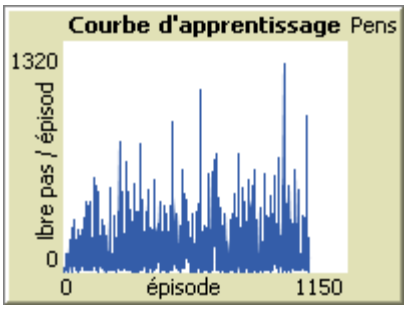


Figure 5.7 - Courbe d'apprentissage pour **r < 0**

5.3.1.2 Influence de l'initialisation de la table d'utilité Q

Etudions maintenant l'influence de l'initialisation de la table d'utilité Q sur l'apprentissage pour cela, nous avons tester le programme avec trois formes d'initialisation de Q ($Q_{init} > 0$, $Q_{init} < 0$ et $Q_{init} = 0$) :

1ier Cas : $Q_{init} > 0$

L'agent effectue un plus grand nombre de pas pour atteindre l'état cible ; il explore constamment l'environnement (figure 5.8).

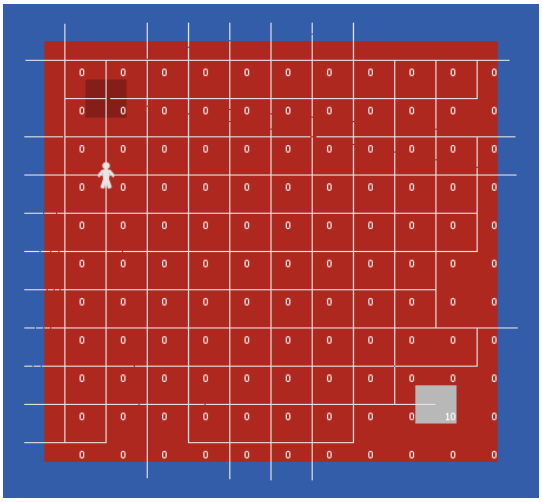


Figure 5.8 - Simulation de la stratégie de l'agent pour $\mathbf{Q_{init} > 0}$

Nous remarquons que :

- ✓ Dés le début, il y a une grande exploration (figures 5.8 et 5.9.b) du labyrinthe
- ✓ Le nombre d'état visité est relativement supérieur (figure 5.9.b).
- ✓ L'agent a visité chaque recoin du labyrinthe et l'état cible a été découvert plus rapidement (figure 5.8)
- ✓ L'utilisation du comportement explorateur favorise l'accélération de l'apprentissage.

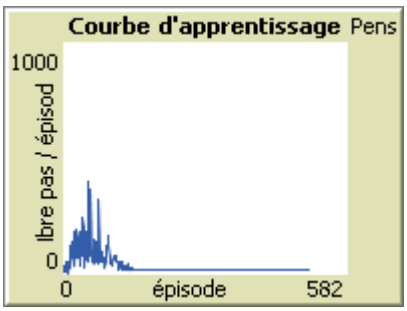


Figure 5.9.a -Courbe d'apprentissage $Q_{init}=0$

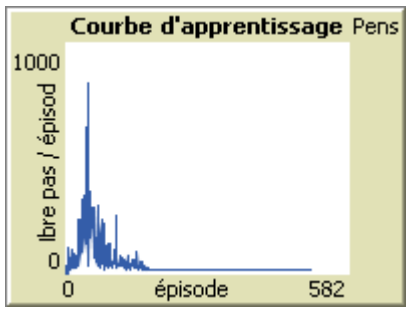


Figure 5.9.b- Courbe d'apprentissage $Q_{init}=10$

Comparaison du nombre de pas par épisode dans les deux cas.

La valeur initiale Q_{init} de la fonction d'utilité influence donc la vitesse de l'apprentissage au début. En effet, si on calcule la première mise à jour de l'équation (3.15), on obtient :

$$Q(s,a) := Q(s,a) + \alpha[\gamma \max_{v \in U} Q(s',v) - Q(s,a)]$$

$$:= Q_{init} + \alpha[\gamma \max_{v \in U} Q_{init} - Q_{init}]$$

$$Q(s,a) := Q_{init} - \alpha(1-\gamma)Q_{init} \qquad (5.1)$$

Pour $Q_{init} > 0$, les états déjà visités auront une valeur d'utilité inférieure à celles des états non visités ($Q(s,a) < Q_{init}$) selon la formule (5.1): Les états non visités seront donc plus attirants ce qui induit une exploration au début de l'apprentissage.

2ième Cas : $Q_{init}0 < 0$

Les états déjà visités auront une valeur d'utilité supérieure à celles des états non visités ($Q(s,a) > Q_{init}$) selon la formule (5.1). Les états visités seront donc plus attirants, ce qui induit une exploration moins grande au début de l'apprentissage. Ce comportement, ralentit considérablement l'apprentissage, donc il est préférable de l'éviter.

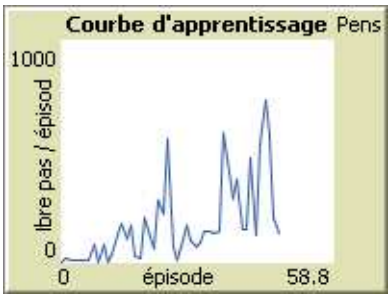


Figure 5.10 - Courbe d'apprentissage Qinit = -1

La figure 5.10 montre que l'épisode ne se termine pas car le l'agent tourne en rond dans une zone de l'environnement où les mises à jour de la table Q converge vers zéro.

3ième cas $Q_{init} = 0$

Prenons maintenant le cas $Q_{init} = 0$, les états déjà visités auront une valeur d'utilité identique à celles des états non visités ($Q(s,a) = Q_{init}$). Le comportement est pleinement aléatoire au début de l'apprentissage. C'est la valeur par défaut la plus utilisé dans les applications de l'apprentissage par renforcement.

Pour conclure, On établi le tableau récapitulatif suivant [Benhamza et *al.*, 2007b, Benhamza et *al.*, 2007d]

Valeur de Qinit	Comportement de l'agent au début de l'apprentissage
$\mathbf{Q_{init} > 0}$	Exploration plus étendue (les états non visités sont plus attirants)
$\mathbf{Q_{init} < 0}$	Exploration limitée. Apprentissage très réduit
$\mathbf{Q_{init} = 0}$	Exploration aléatoire

Tableau 5.1- Comportement de l'agent en début de l'apprentissage pour les différentes initialisations de la table d'utilité Q

Conclusion

Si la cible est éloignée et si on veut avoir un comportement d'exploration au début pour accélérer l'apprentissage, il faut alors initialiser la fonction d'utilité à une valeur positive.

5.3.2 Cas général : Labyrinthe avec obstacle

Nous étudions ici, le cas plus général où les transitions qui mènent à l'état cible ont une récompense « r », elles ont une autre récompense « r'» si elles mènent à un obstacle et nulle ailleurs.

$$R(s,a,s') = \begin{cases} r & \text{si } s' = \text{état but} \\ r' & \text{si obstacle} \\ 0 & \text{sinon} \end{cases}$$

C'est le cas de l'agent qui évolue dans un labyrinthe avec mur (Figure 5.11). Il a le choix de se déplacer dans les quatre directions cardinales (droite, gauche, haut et bas). Si une action entraîne l'agent dans un mur, celui-ci reste dans son état courant.

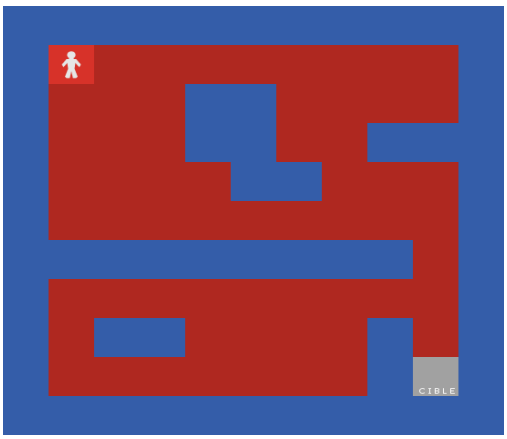


Figure 5.11 - Labyrinthe avec murs

5.3.2.1 Influence de l'initialisation de la table d'utilité Q

Etudions l'impact de l'initialisation de la table Q au début d'apprentissage :

1ier cas : si l'agent ne rencontre pas d'obstacle : Cas traité précédemment (5.3.1.2)

2ième cas : Si l'agent rencontre un mur :

Si nous appliquons le même principe vu précédemment (5.3.1.2) avec l'équation (1.6) :

$$Q(s,a) = Q_{init} - \alpha\,[(1-\gamma)Q_{init} - r']$$

$$Q(s,a) = Q_{init} - \alpha\,(1-\gamma)\,[Q_{init} - r'/(1-\gamma)]$$

Nous remarquons que la borne n'est plus zéro mais $(r'/1-\gamma)$.

Ainsi,

- ✓ $Q_{init} > r'/1-\gamma$ induit une exploration systématique, (figure 5.12)
- ✓ $Q_{init} < r'/1-\gamma$ induit un ralentissement de l'apprentissage au début.
- ✓ et $Q_{init} = r'/1-\gamma$ induit un comportement tout à fait aléatoire

La stratégie de l'agent est schématisée sur les figures (5.12 et 5.13) :

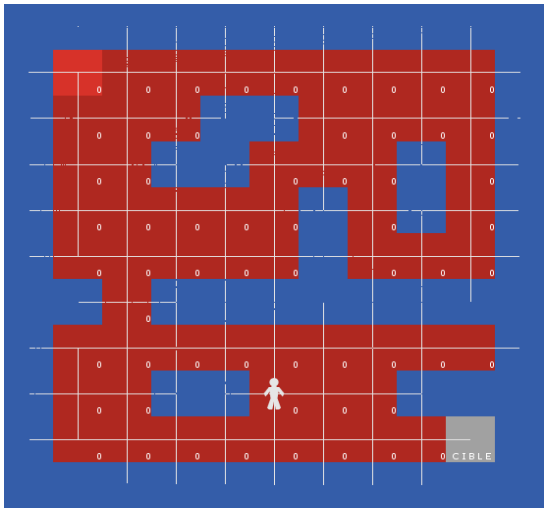


Figure 5.12 - **Simulation de la stratégie: Exploration étendue**

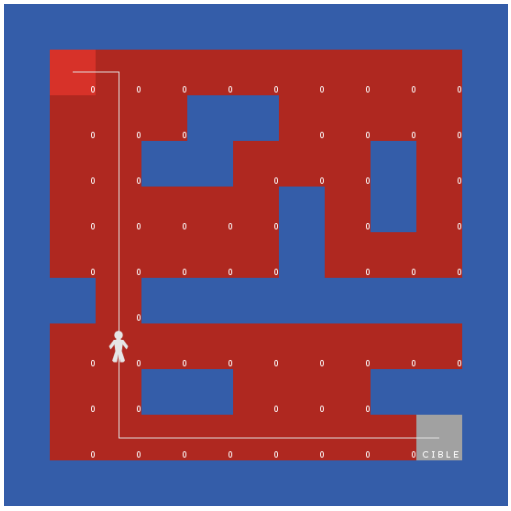


Figure 5.13 - **stratégie optimale**

Le tableau 5.2 synthétise ces résultats pour permettre de choisir le comportement de l'algorithme en début d'apprentissage :

Qinit > r' /1-γ	**Qinit** = r'/1-γ	**Qinit** < r'/1-γ
Exploration systématique et attraction locale du but	Aléatoire	évitement du but

Tableau 5.2 - **Stratégie optimale obtenue au début de l'apprentissage pour différentes initialisations de la table d'utilité Q.**

5.4 Cas des récompenses calculées par une fonction objective « estimateur de progrès »

Pour accélérer l'apprentissage, certains auteurs définissent les récompenses comme un critère lié à une fonction objective [Mataric, 1994]. Cette fonction définit une mesure de chaque couple état-action par rapport à un objectif.

Si f une fonction définie de l'espace état-action sur R. Le but de l'apprentissage revient à maximiser cette fonction f(s,a).

5.4.1 Cas de fonction distance

Dans notre cas (navigation dans un labyrinthe), cette fonction peut être la distance de Manhattan de l'état actuel à l'état but : **F(s,a)=d(s,but).**

Si l'état « s » a pour coordonnées (x_1, y_1) et l'état cible a pour coordonnées (x_2, y_2), alors la distance de Manhattan est $d=|x_1 - x_2| + |y_1 - y_2|$. L'objectif de l'agent est alors de minimiser cette fonction.

Mais on peut aussi poser les hypothèses suivantes : la fonction récompense vaut 1 si l'action « a » rapproche l'agent de l'état cible. Elle vaut zéro si l'agent ne bouge pas et -1 si l'agent s'éloigne de l'état cible. La fonction de récompense est recalculée à chaque action de l'agent.

Dans les deux cas, l'agent est de moins en moins punit lorsqu'il s'approche de l'état cible. La stratégie globale est le plus court chemin vers l'état but.

5.4.1.1 Pour le cas de labyrinthe sans mur

Avec ces récompenses, les résultats de l'apprentissage sont évidemment meilleurs. Les épisodes sont plus rapides et l'algorithme converge en 15 épisodes. Pour ce cas relativement simple, cette technique de calcul de récompense est très favorable.

5.4.1.2 Pour le cas de labyrinthe avec mur

L'exemple précédent est extrêmement simple. Prenons maintenant le cas d'un labyrinthe avec quelques murs au milieu.

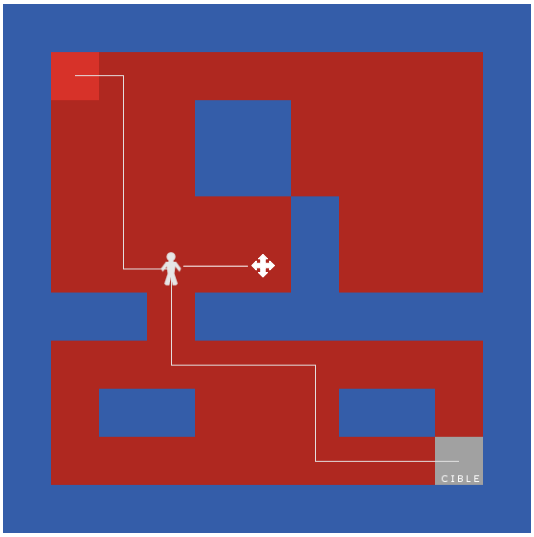


Figure 5.14 - 1ier cas

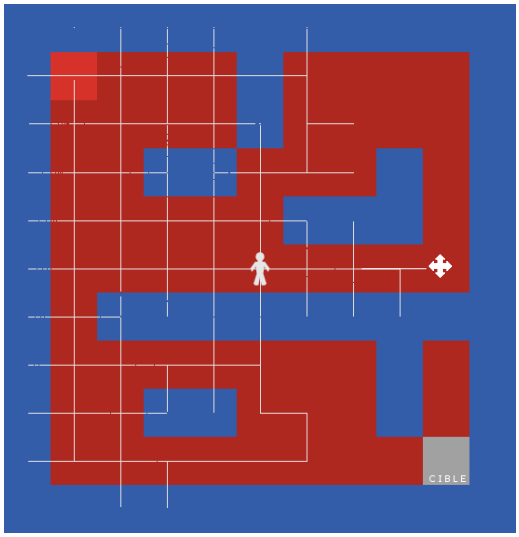


Figure 5.15 - 2ième cas

Simulation de la stratégie de l'agent (Existence d'une impasse)

La stratégie optimale pour atteindre l'état cible est décrite par les deux cas de figure 5.14 et 5.15, on remarque que l'agent se bloque dans une impasse qui le rapproche de l'état cible au sens de minimum de distance entre les deux états (état courant et cible). Cette forme de récompenses est donc trompeuse pour l'agent.

La fonction « objective » ou « estimateur de progrès » n'est plus adaptée et induit en erreur car l'algorithme met plus de temps à converger, mais surtout il se bloque dans le coin formé par les murs.

Il est possible pour l'agent de sortir de cette impasse mais pour cela, ce dernier doit revenir en arrière ce qui est équivaut à choisir une valeur d'utilité Q(s,a) moins optimale. Cela est possible si seulement plusieurs actions d'exploration se succèdent c'est-à-dire très rarement.

Donc

✓ Les méthodes avec fonction « Distance » sont donc risquées car les murs peuvent créer des situations complexes (une impasse) où l'agent peut se bloquer. Il est donc préférable d'utiliser ces approches avec prudence dans la mesure où elles peuvent conduire à un comportement néfaste [Benhamza et *al.*, 2007b, Benhamza et *al.*, 2007d].

5.4.2 Cas de plusieurs récompenses hétérogènes

Les fonctions de récompense hétérogènes joignent un renforcement à l'accomplissement de chaque but et sous-buts avec l'application de comportements associés. Dans ce cas idéal, le renforcement est immédiat et significatif en fournissant la récompense ou la punition en réponse aux événements reçus, c'est ce qu'on appelle «Estimateur de progrès » [Mataric, 1997].L'exemple de récompense ci-dessous, illustre chacune des fonctions de récompenses hétérogènes fournit à une partie de la structure de la tâche.

Une fonction hétérogène générale de récompense a la forme suivante :

$$R = \begin{cases} r_1 \text{ si event1} \\ r_2 \text{ si event 2} \\ \quad . \qquad . \\ \quad . \qquad . \\ r_n \text{ si event n} \\ 0 \text{ sinon} \end{cases}$$

L'estimateur de progrès utilise la connaissance de domaine pour mesurer le progrès durant l'apprentissage. Tandis que le renforcement immédiat n'est pas disponible dans beaucoup de domaines, le renforcement intermédiaire peut être fourni en estimant le progrès de l'agent relativement à son but et en affectant la récompense en conséquence. Ceci fournit une source continue de renforcement avec la connaissance implicite du domaine. Il permet de d'accélérer l'apprentissage et diminuer la fragilité de l'algorithme d'apprentissage de manière en encourager le comportement d'exploration vers le but.

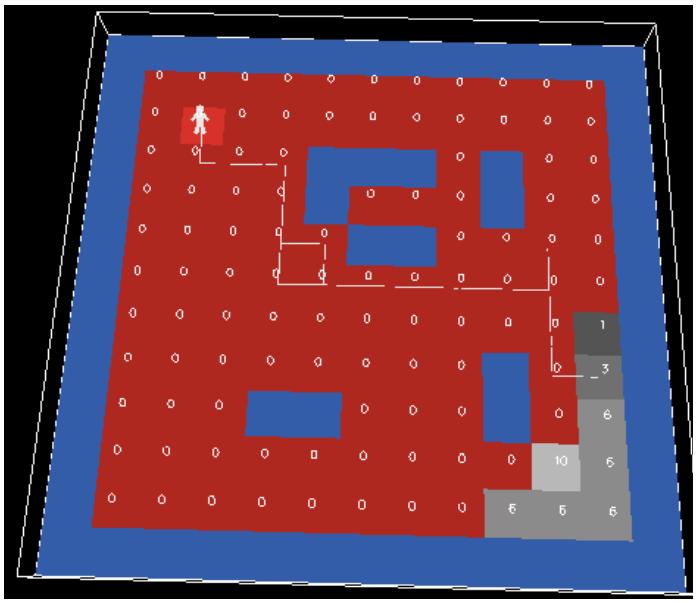


Figure 5.16 - Simulation de la stratégie de l'agent
Cas de plusieurs récompenses hétérogènes

Donc

- ✓ L'utilisation des fonctions de récompense hétérogènes et d'estimateurs de progrès accumulée avec la connaissance de l'environnement peut rendre de ce fait l'apprentissage plus efficace et plus rapide vers des buts associés, à la différence d'une simple fonction monolithique de renforcement.

Nous terminons cette partie expérimentale par étudier l'influence des deux paramètres d'apprentissage α et γ .

5.5. Etude de l'Influence de la valeur des paramètres α et γ

5. 5.1 Pour le paramètre α

En théorie, α représente le taux d'apprentissage et doit décroître à chaque nouveau passage par l'état « s » et l'action « a » mais dans la pratique, et pour éviter de stocker des informations supplémentaires, on utilise un coefficient fixe.

La valeur la plus courante pour différents utilisateurs et auteurs est 0,1. Quelque fois, ces derniers recommandent de fixer α à 1 pour aller le plus vite possible.

Pour notre application, nous avons testé les deux valeurs et nous remarquons que lorsque le nombre d'épisode est initialisé 100 avec α=0,1 on n'atteint pas la stratégie optimale (Figure 5.17).

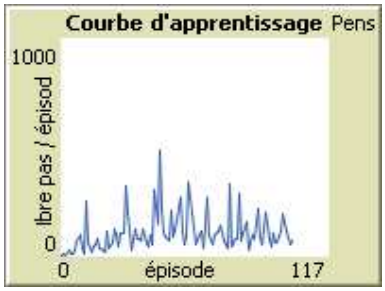


Figure 5.17 - Courbe d'apprentissage du Q-Learning sur l'exemple du labyrinthe Evolution du nombre de pas par épisode en fonction du nombre d'épisode pour r =10 γ= 0,9 **α=0,1** et Qinit=0 Nombre épisode=100

Mais si on initialise α=1 avec le même nombre d'épisode 100, on voit que l'apprentissage se fait normalement et on aboutit finalement à la stratégie optimale.

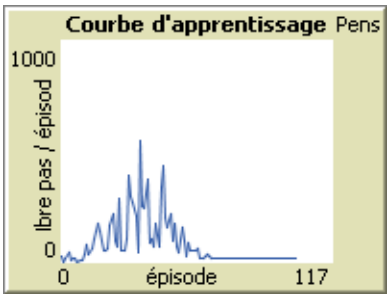


Figure 5.18 - Courbe d'apprentissage du Q-Learning sur l'exemple du labyrinthe Evolution du nombre de pas par épisode en fonction du nombre d'épisode pour r =10 γ= 0,9 α=1 et Qinit=0 Nombre épisode=100

Pour un nombre d'épisodes supérieur ou égal à 1000, la stratégie optimale est atteinte pour les deux cas mais plus rapidement avec α =1 (Figure 5.18).

Donc

✓ le paramètre α influence légèrement la vitesse de convergence mais il peut s'avérer utile de réaliser des tests pour choisir la meilleure valeur de α selon l'application et la complexité du problème.

5.5.2 Pour le paramètre γ

Le paramètre γ permet normalement de donner plus ou moins d'importance aux récompenses éloignées dans le futur. Sa valeur la plus courante est 0,9. En fait, ce paramètre n'a pas d'influence que si on a plusieurs cibles à atteindre.

Prenons l'exemple particulier de la figure 5.19 où on a plusieurs cases qui incitent une seule fois à la même récompense.

Position de l'agent

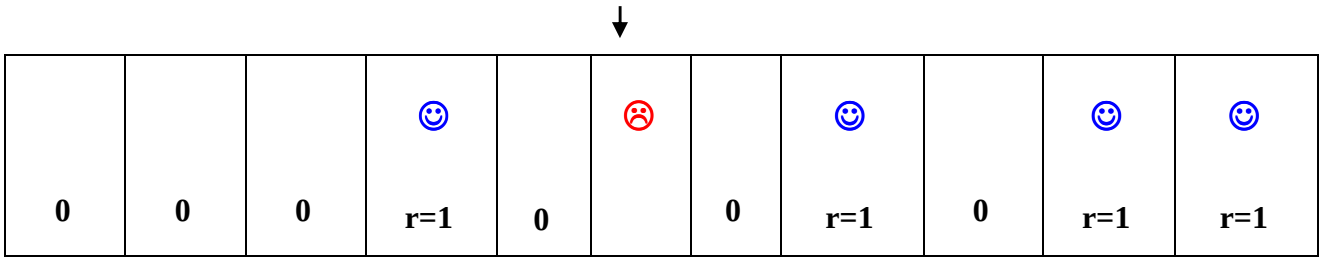


Figure 5.19 - Exemple de configuration où la valeur de γ intervient sur le comportement de l'agent

L’agent a le choix entre deux trajectoires, la première en commençant par la récompense la plus proche (à gauche), la seconde en commençant par la zone la plus dense en récompenses (à droite).

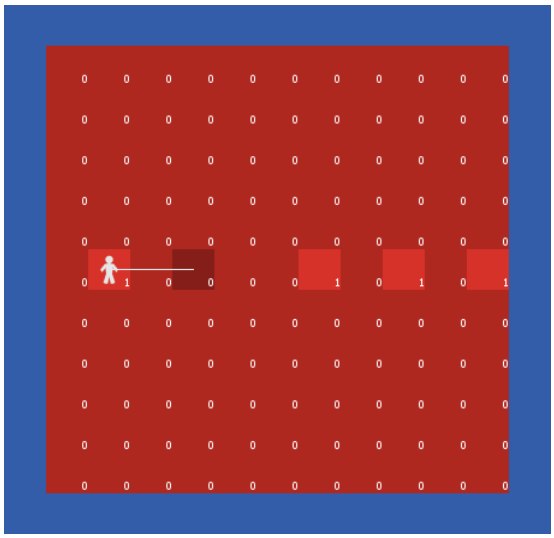


Figure 5.20 - Simulation de la stratégie de l’agent : avec γ=0,1 l’agent opte pour l’action gauche

Dans cet exemple, l'espérance de gain de l’action «gauche » s'écrit

$$Q(G) = \gamma^2 + \gamma^7 + \gamma^9 + \gamma^{11} .$$

et l'espérance de gain de la commande « à droite » vaut

$$Q(D) = \gamma^3 + \gamma^5 + \gamma^7 + \gamma^{16}.$$

Si $\gamma = 0,1$ alors $Q(G) = 0,0100001 > Q(D) = 0,0010101$ et l’agent prendra la première trajectoire (à gauche) délaissant les récompenses éloignées dans le futur.

En revanche, si $\gamma = 0,8$, alors $Q(G) = 1,06983227 < Q(D) = 1,0775427$ et l’agent prendra la seconde trajectoire (à droite) délaissant la récompense proche.

Donc

✓ Si on choisit γ faible, alors l’agent sera plutôt opportuniste et atteindra les récompenses proches, si on choisit γ proche de 1, alors l’agent préférera atteindre des zones plus denses en récompenses même éloignées.

5.6 Conclusion

Dans ce chapitre, nous avons entrepris une série de tests avec différentes valeurs initiales pour les paramètres d'apprentissage dans l'application de navigation d'un agent dans un labyrinthe crée aléatoirement.

Les résultats expérimentaux sur ce dernier, nous ont permis de mettre en évidence l'importance du choix des valeurs initiales de chacun des facteurs α, γ, r, la table utilité Q dans la stratégie adoptée par l'agent.

Finalement, ces travaux nous ont aidé à finaliser la conception d'un agent par Q-learning et à élaborer ainsi, des règles pour une initialisation conforme de ses valeurs.

Mise en oeuvre

Conclusion

Cette partie nous a permis de valider nos travaux, avec le modèle de conception de l'agent intelligent que nous avons élaboré, et confirmé sur une application de robotique mobile. Avec des tests de simulation, nous avons pu offrir notre apport dans ce domaine avec une spécification rigoureuse des différents paramètres d'apprentissage selon le comportement désiré.

Nous terminons, ce travail en donnant la conclusion générale et les perspectives souhaitées.

Conclusion générale et perspectives

Conclusion générale et perspectives

Le point de vue que nous avons adopté au cours de ce mémoire pour la conception de notre système est celui qui agit de manière intelligente et rationnelle, système que l'on a décidé de désigner par le terme "agent".

Pour automatiser le processus de conception d'agent intelligent, nous avons proposé, donc, d'utiliser les algorithmes d'apprentissage par renforcement. Ces derniers constituent assurément une technique privilégiée pour la modélisation de comportements adaptatifs intelligents.

En tant que méthode d'apprentissage, elle permet à un agent d'ajuster son comportement au cours d'expériences successives en extrayant de l'information de ses interactions avec son environnement. Étant non supervisée, elle permet de ne pas violer la contrainte d'autonomie et de réactivité de l'agent qui apprend, le signal de renforcement fourni, de façon naturelle par l'environnement.

Les résultats expérimentaux avec notre modèle de conception appliqué sur une plate-forme de robotique mobile autonome ont été très encourageants et ont permis de présenter sa certification.

Aussi, avec une large gamme de tests de simulations basée sur une étude formelle, nous avons pu spécifier le comportement de notre agent selon les différentes initialisations des paramètres d'apprentissage.

Nous sommes arrivés à définir leurs valeurs spécifiques permettant non seulement d'accélérer l'apprentissage c'est à dire la vitesse de convergence qui est un problème majeur pour les algorithmes d'apprentissage par renforcement mais aussi d'obtenir les meilleures stratégies.

Le choix de la fonction de renforcement et des valeurs initiales de la table d'utilité ainsi que celui des facteurs d'apprentissage a un impact majeur sur la performance de ces algorithmes. Ce choix n'est pas trivial et doit être fait en accord avec le comportement désiré.

Notamment, certaines valeurs peuvent amener l'agent à avoir un comportement néfaste qui doit être évité.

Notre étude a abouti d'une part à une critique des travaux réalisés dans cet axe et d'autre part à l'élaboration de règles pour une estimation correcte des différentes valeurs initiales des paramètres d'apprentissage.

Les résultats expérimentaux satisfaisants ont bien montré l'efficacité de notre travail et ont permis ainsi de finaliser la conception d'agent intelligent. Cet agent sera capable, non seulement, d'apprendre plus vite en améliorant sa vitesse de convergence, mais aussi d'éviter les comportements néfastes.

Le thème abordé dans ce travail de recherche, et plus précisément les travaux de simulations qui ont été conduit dans ce mémoire, amènent à des perspectives assez diverses telles que :

- ✓ **L**a construction d'un modèle fondé sur un historique d'observations et d'actions afin d'optimiser la fonction d'utilité ;
- ✓ **L**'utilisation d'un cadre multi agents qui rend l'environnement non stationnaire et dont il est nécessaire de coordonner les buts de tous les agents pour satisfaire l'objectif global;
- ✓ **L**e traitement des algorithmes dans un cadre réel pour représenter de façon plus naturelle l'évolution de grandeurs continues et les observations partielles de l'état courant de l'agent.
- ✓ **E**nfin, le travail présenté dans ce mémoire s'est essentiellement appuyé sur des applications liées à la robotique mobile. Nous pensons toutefois que notre travail n'est pas restreint à ce cadre. Nous souhaitons donc étendre l'applicabilité de notre approche à des projets plus divers.

Finalement, ce travail de recherche nous motive à penser que l'apprentissage par renforcement représente une voie prometteuse pour la commande de système complexe dont on ne connaît pas le comportement et notamment dans les domaines où l'humain ne peut intervenir directement.

Bibliographie

Bibliographie

[Agha, 1986] G. Agha. Actors: A model of concurrent computation in distributed systems.The MIT Press Cambridge MA, 1986;

[Agha et *al*., 1988] G. Agha and C. Hewitt Concurrent programming using actors. In Y. Yonezawa and M. Tokoro, editors, Object-Oriented Concurrent Programming. MIT Press, pp. 37-57, 1988.

[Agha et *al*, 1993] G.Agha, P. Wegner and A. Yonezawa, Research Directions in Concurrent Object-Oriented Programming. The MIT Press: Cambridge, MA, 1993.

[Albus, 1975] J. S. Albus, A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC) Journal of Dynamic Systems, Measurement and Control, 37(3), pp. 220-227, 1975.

[Asada et *al*, 1991] H. Asada and S. Liu, Transfer of Human Skills to Neural Net Robot Controllers. In Proceeding of IEEE conference on Robotics and Automation, Sacramento, IEEE 1991.

[Arkin, 1998] R. Arkin Behavior- Based Robotics, The MIT Press. 1998

[Baquiast, 2001] J-P Baquiast, A.Cardon, C.Jacquemin, Les automates intelligents robotique, vie artificielle, réalité virtuelle La revue mensuelle N°21 , 2001

[Baxter, 1998] J.Baxter, A. Tridgell, and L.Weaver Knightcap: a chess program that learns by combining TD(λ) with game-tree search. In International Conference on Machine Learning, volume 15, 1998.

[Baxter et *al*., 1999] J. Baxter and P. Bartlett, Direct Gradient-Based Reinforcement learning: I. Gradient estimation, algorithms, technical report, research school of information sciences and engineering, Australian national university, 1999

[Behnke, 2005] S. Behnke et M. Bennewit, Learning to play soccer using imitative reinforcement . In Proc. Of the ICRA Workshop on social aspects of robot programming through demonstration, Barcelona, April 2005

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

[Benhamza et *al.*, 2007a] K. Benhamza, H. Seridi et H. Akdag, Conception d'agents intelligents avec apprentissage par renforcement dans un environnement de grande de taille, Journées Ecole Doctorale et équipe en réseaux, JED'07 Annaba Algérie 27-28 mai 2007

[Benhamza et *al.*, 2007b] K. Benhamza, H. Seridi et H. Akdag, *Conception d'Agents Intelligents avec Apprentissage par Renforcement*, Proceedings of 3èmes journées internationales sur l'informatique graphique, Constantine, Algérie, pp 250-257, Octobre 2007.

[Benhamza et *al.*, 2007c] K. Benhamza, H. Seridi et H. Akdag Conception d'Agents Intelligents avec Apprentissage par Renforcement dans un environnement de grande de taille, Proceedings Séminaire national sur le langage naturel et l'intelligence artificielle LANIA'2007, Chlef Algérie, pp 155-162, 20-21 Novembre 2007.

[Benhamza et *al.*, 2007d] K. Benhamza, H. Seridi et F. Nouar, Accélération du processus d'apprentissage d'agent Intelligent conçu par Q-learning. Colloque International sur les Equations aux Dérivées Partielles et leurs Applications « CISEDPA'07 » Guelma Algérie 05-07 Novembre 2007.

[Bernstein et *al.*, 2001] D. Bernstein, S.Zilberstein , R. Washington, et J. Bresina, Planetary rover control as a markov decision process. In The 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, Montreal, Canada. 2001.

[Bertsekas, 1989] D. Bertsekas et J. Tsitsiklis. Parallel distributed computation. In Numerical Methods, Prentice Hall, Englewood Cliffs, NJ, 1989.

[Bertsekas, 1995] D. Bertsekas Dynamic programming and optimal control, Vol. 1 & 2. Athena Scientific, Belmont, Massachusetts, 1995.

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

[Bouzid, 2001] M. Bouzid Contribution à la modélisation de l'interaction agent/environnement modélisation stochastique Thèse de doctorat de l'université Henri Poincaré Nancy, 2001

[Boyan, 1993] J. Boyan and M. Littman Packet routing in dynamically changing networks: A reinforcement learning approach. In J. Cowan, G. Tesauro, and J. Alspector, editors, Advances in Neural Information Processing Systems 6. Morgan Kaufmann, 1993.

[Buffet, 2000] O. Buffet Apprentissage par renforcement dans un système multi-agents : rapport de stage DEA, Informatique Nancy : UFR STEMIA, Université H. Poincaré- Nancy1 2000.

[Buffet, 2003] O. Buffet Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatif Thèse UFR STEMIA Nancy1 2003.

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

[Bratman et *al*, 1988] M. Bratman, Israel and Pollack M. Plans and Resource-Bounded Practical Reasoning computational Intelligence, 4:349-355 1988.

[Brooks, 1986] R.A Brooks. A robust layered control system for mobile robot. IEEE Journal of robotics and automation, 2(1): 14-23, 1986

[Brooks, 1991] R.A Brooks Intelligence without representation. Artificail intelligence, 47 :139-159, 1991

[Cabarbaye et *al.*, 1999] A. Cabarbaye F. Garcia L. Tomasini Apport de l'apprentissage par renforcement aux problèmes de maintenance optimale : application aux constellations de satellites, Congrès ROADEF ' 99.

[Cardon et *al.*, 2001] S.Cardon, A. Mouaddib, S. Zilberstein et R. Washington Adaptive control of acyclic progressive processing task structures. In Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI-2001, pages 701-706, 2001

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

[Chaib-Draa, 2002] Chaib-draa B., Demolombe R, "L'interaction comme champ de recherche", Information-Interaction-Intelligence, Cépaduès, Toulouse, Numéro hors série sur l'interaction, 2002

[Coulom, 2002] Coulom R, Apprentissage par renforcement utilisant les réseaux de neurones, avec des applications au contrôle moteur", Ph.D. thesis, INPG, Grenoble, 2002.

[Dayan, 1992] Dayan P. The convergence of TD(λ) for general λ. Machine Learning, 8(3):341-362, 1992.

[Dayan et *al.*, 1994] P. Dayan And T.J.Sjnowski TD(λ) converges with probability 1. Machine Learning, 14, 295-301,1994

[Desjardins, 2007] C. Desjardins, Approximation de fonctions de valeurs DAMAS, université LAVAL, séminaire hiver, 2007

[Demazeau, 1995] Y.Demazeau, From Interactions to Collective Behaviour in Agent-Based Systems, Proceeding of the First European Conference on Cognitive Science, Saint-Malo, 1995

[Drogoul, 1999] A.Drogoul et J-A. Meyer, Intelligence artificielle située. Actes du colloque IAS'99, Hermès, Paris, France 1999.

[Drogoul, 2001] A. Drogoul, D. Servat "Informatique diffuse: vers un monde fait d'agents?" Actes des JFIADSMA'01 Hermès, Paris, France; 2001.

[Dutech, 1999] A. Dutech Apprentissage d'environnement : approches cognitives et comportementales. Thèse de doctorat, Ecole Nationale supérieure de l'aéronautique et de l'espace, Toulouse, février 1999.

[Eli et *al.*, 2001] L. Elie et B.Lapeyre Introduction aux méthodes de Monté Carlo. Cours, Paris CERMICS, ENPC, 2001.

[Fabiani et *al.*, 2001] P. Fabiani, J.-P. Forges, et F. Garcia. Décision dans l'Incertain. 2001.

[Ferber, 1995] Jacques Ferber. Les systèmes multi-agents vers intelligence collective. interEditions, 1995.

[Ferber, 1997] Jacques Ferber. Les systèmes multi-agents : Un aperçu général. TSI : Techniques et Sciences Informatiques, 1997.

[Ferguson, 1992] D. Ferguson Bit-tree, a data structure for fast file processing. CACM, 35(6): 114-120, 1992.

[Florea et *al.*, 2002] Adina Florea, Daniel Kayser, Stephan Pentiuc, Agents Intelligents, Ch.6 Politechnica University of Bucharest-2002. http://turing.cs.pub.ro/auf2/home.html

[Franklin et Graesser, 1996] S. Franklin, et A. Graesser, Is it an agent, or just a program?: A taxonomy for autonomous agents Proceeding of third international workshop Agent theories, architectures and langages, page 21-35 Springer-Verlag, 1996.

[Garcia, 1997] F. Garcia Apprentissage par renforcement en horizon fini I Comparaison du Q-learning et du R-learning Revue READ 1997

[Garcia, 2004] P. Garcia Exploration guidée et induction de comportement génériques en apprentissage par renforcement. PHD thesis , INSA de rennes, 2004

[Gérard, 2002] P. Gérard Systèmes de classeurs : étude de l'apprentissage latent. Thèse de Doctorat de l'Université Paris 6. Spécialité Informatique, 2002

[Girard-Faugère, 1997] M. Girard-Faugère ADAM, un gestionnaire d'applications réparties en environnement dynamique. Thèse de Doctorat de l'Université Paris VI, 1997.

[Goldman, 1996] Goldman C.V., Learning in multi-agent systems in proceedings of the 13th National conference on artificial intelligence (AAAI 96) Cambridge, united Kingdom, 1996 MIT Press

[**Harmon, 1996]** M. E. Harmon Reinforcement learning: A tutorial. http ://www.citeseer.nj.nec.com/harmon96reinforcement.html, 1996.

[Hebb, 1949] D. O. Hebb, The Organization of Behavior. New York: John Wiley, 1949

[Hewitt et *al.* 1973] C. Hewit P. Bishop et R. Steiger. A universal modular actor formalism for artificial intelligence. IJCAI-73 , 1973.

[Horvitz et *al.*, 1988] E. J. Horvitz, J.S. Breese, et M. Henrion. Decision theory in expert systems and artificial intelligence. International Journal of Approximate Reasoning, 2 :247–302, 1988.

[Hopfield, 1982] J.J Hopfield, Neural networks as physical systems with emergent computational abilities. Proceedings of the National Academy of Science of USA, 79, 2-554. 1982

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

[Jaakkola et *al*, 1994] T. Jaakkola, M.I Jordan and S. Singh On the convergence of stochastic iterative dynamic programming algorithms. Neural Computation, 6(6), November 1994.

[Jars, 2005] Contribution des sciences sociales dans le domaine de l'intelligence artificielle distribuée: Alone, un modèle hybride d'agent apprenant, Thèse de doctorat, université C.B- Lyon 1, 2005.

[Jenning et al.,1998] N. Jennings, K. Sycara et M. Wooldridge. A roadmap of agent research and development Journal of autonomous agents and multi-agent systems, 1:275-306, 1998.

[Jozefowiez, 2001] J. Jozefowiez, Conditionnement opérant et Problèmes décisionnels de Markov. Thèse de doctorat de l'Université Lille III, Lille 2001.

[Kaelbling et al., 1996] L. P. Kaelbling and M. L. Littman. Reinforcement learning : A survey. Journal of Artificial Intelligent Research, 4: 237-277. may 1996.

[Kaelbling et al., 1998] Kaelbling, L. P., Littman, M. L. et Cassandra, A. R. Planning and acting in partially observable stochastic domains. Artificial Intelligence, 101:99-134. 1998.

[Kimura, 1997] Kimura H. and Kobayashi S. Reinforcement learning for locomotion of a two-linked robot arm. In Proceedings of the 6th European Workshop on Learning Robots, pages 144{153, 1997.

[Laroche, 2000] Laroche P., Processus Décisionnels de Markov appliqués à la planification sous incertitude. Thèse de doctorat, Université Henri Poincaré Nancy 1. 2000.

[Lin, 1992] Long-Ji Lin, Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching, Machine Learning 8:293-321(1992),

[Luck et al. ,2004] Luck M, Ashri R. and d'Inverno M., Agent-Based Software Development, Artech Hourse, 2004

[Maes, 1994] P. Maes Agents that reduce work and information overload. Communications of the ACM, 37(7): 31-40, Juillet 1994

[Mataric, 1994] M.J Mataric, Reward Functions for accelerated learning. In Proceeding of the 11th ICML, pages 191-189, 1994

[Mataric,1997] M.J. Mataric, Reinforcement learning in the multi-robot domain. Autonomous Robots, 4(1):73{83, 1997.

[Mitchell, 1997] T. Mitchell Machine Learning. McGraw Hill, 1997

[Müller, 1997] Müller J.P, Control architectures for autonomous and interacting agents: A survey. In Intelligent agent systems: Theoretical and Practical Issues, lecture Note in AI, editeur L. Cavedon, A. rao et W. Wobcke, volume 1209, Springer, 1997.

[Munos, 1997] R. Munos L'apprentissage par renforcement Etude du cas continue PHD Thesis Cemagref, Ecole des Hautes Etudes en sciences sociales. 1997

[Munos, 2007] R. Munos, Transparents cours "Apprentissage par Renforcement", Master Mathématiques, Vision, Apprentissage, ENS Cachan. http://sequel.futurs.inria.fr/munos/master-mva

[Ndiaye, 1999] Ndiaye S. Apprentissage par renforcement en horizon fini : application à la génération de règles pour la Conduite de Culture. Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, février 1999.

[Newell, 1982] The knowledge level. Artificial Intelligence, 18:87-127, 1982

[Odell, 2002] J.Odell Objects and agents compared. Journal of object technology 1(1) :41-53, 2002

[Odell et al , 2002] Odell J., Parunkak H., Fleischer M., and Brueckner S., Modeling agents and their environment. Lecture Notes on Computer Science, 2585: 16-31, 2002

[Pavlov, 1927] Pavlov, I. P. Conditioned reflexes Oxford, UK: Oxford University Press (1927).

[Pfeifer et al., 1999] R. Pfeifer, et C. Scheier. Understanding Intelligence. Cambridge, Mass.: MIT Press. 1999

[Piaget, 1967] J.Piaget La Psychologie de l'Intelligence. Armand Colin, 1967.

[Potapov et al., 2003] A.B. Potapov, M.K. Ali. Convergence of reinforcement learning algorithms and acceleration of learning . Phys. Rev. E, 67, 026706 (2003).

[Puterman, 1994] M. L. Puterman. Markov Decision Processes–Discrete Stochastic Dynamic Programming.John Wiley and Sons, Inc., New York, USA, 1994.

[Puterman, 2005] M. L. Puterman,. Markov Decision processes : discrete stochastic dynamic programming. Wiley-Interscience, New York, deuxième édition. 2005

[Rescorla et al., 1972] R.A. Rescorla et A.R Wagner A theory of pavlovian coditionning: variation in the effectiveness of reinforcement and non reinforcement Edité dans Black, A, H & Prokazy, W. F., editeurs, Classical Conditionning II, pages 64-99 Appleton Century Croft, New York, 1972

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

[Rouanet, 1964] H. Rouanet Les modèles stochastiques d'apprentissages, Mathématiques et sciences humaines, tome 5 Pages 3-10, 1964

[Richard, 2001] N. Richard Description de comportements d'agents autonomes évoluant dans des mondes virtuels PHD Thesis, ENST, Paris.

[Rubinstein, 1981] R.Y. Rubinstein, Simulation and the Monte Carlo Method. Wiley, NY1981.

[Russel et Norvig, 95] S. Russel , Norvig P. Artificial intelligence: A modern Approach. Prentice – Hall, Englewood Cliffs, NJ, 1995

[Russel et Norvig, 2003] S. Russell, et Norvig, P. (2003). Artificial Intelligence: A Modern Approach, Prentice Hall Series 2003

[Sallantin, 1997] J. Sallantin, Les agents intelligents: essai sur la rationalité des calculs, Hermès, 1997

[Samuel, 1959] A. Samuel Some studies in machine learning using the game of checkers. IBM journal of research development, 3(3) :210{229, 1959.

[Sehad et al., 1995] S. SEHAD and C. TOUZET, "Neural Reinforcement Path Planning for the Miniature Robot Khepera," WCNN'95, Washington D.C., USA, 17-21 July 1995

[Schaal, 1994] S. Schaal and C. Atkeson. Robot juggling: An implementation of memory basedlearning. Control System Magazine, 1994.

[Sehad, 1996] S. Sehad Contribution à l'étude et au développement de modèles connexionnistes à apprentissage par renforcement : application à l'acquisition de comportements adaptatifs. / Thèse de doctorat. Université de Montpellier II, 1996. – 112 - 1996

[Sigaud, 1999] O. Sigaud, and P.Gérard, Contribution au problème de la sélection de l'action en environnement partiellement observable. Intelligence Artificielle Située Hermès, publisher. Pages 129-146. Paris, France(1999).

[Sigaud, 2004] O. Sigaud, Comportements adaptatifs pour des agents dans des environnements informatiques complexes. . Habilitation à Diriger des Recherches. Université Pierre et Marie Curie, Paris 6. Pages 27-47, 2004.

[Simmons et al., 1995] R.Simmons, et S. Koenig, Probabilistic robot navigation in partially observable environments. In Proceedings of the International Joint Conference on Artificial Intelligence, pages 1080-10871995.

[Sutton, 1988]. R.S Sutton Learning to predict by the method of temporal differences. Machine Learning, 3:9-44. 1988

[Sutton et *al.* 1998] R. S. Sutton and A. G. Barto. Introduction to reinforcement learning. MIT Press /Bradford Books, 1998.

[Shoham 1993] Y. Shoham, « Agent-oriented programming », Artificiel Intelligence, 60(1) :51-92, 1993

[Skinner, 1938] B. F. Skinner, The Behavior of organisms. Appleton – Century Crofts Inc., New York, 1938.

[Tesauro, 1994] G. J. Tesauro TD-Gammon, a self-teaching backgammon program, achieves master-level play. Neural Computation, 6(2) :215{219, 1994.

[Thomas, 2005] V. Thomas, Proposition d'un formalisme pour la construction automatique d'interactions dans les systèmes multi-agents réactifs, thèse université UHP Nancy 1,2005.

[Thorndike,1911] E. Thorndike, L. animal intelligence. MacMillan company, New York, 1911

[Thurn, 1992] S B. Thrun. The role of exploration in learning control. In David A. White and Donald A. Sofge, editors, Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. Van Nostrand Reinhold, New York, NY, 1992.

[Touzet , 1990] C.Touzet Contribution à l'étude et au développement de modèles connexionnistes séquentiels de l'apprentissage, Thèse de doctorat University of Montpellier II. 1990

[Touzet, 1998] C.Touzet Des réseaux de neurones artificiels à la robotique coopérative, HDR. . Habilitation a Diriger des Recherches, 1998

[Tyrrell, 1993] T. Tyrrell Computational mechanisms for action selection, PHD Thesis, university of Edinburgh, 1993

[Uribe et *al*., 1999] A.P Uribe et E.A Sanchez comparison of reinforcement learning with eligibility traces and integrated learning, planning and reacting. Concurrent systems engineering series vol. 54 Ios press, Amsterdam, 1999, pp 154-159

[Vercouter et *al*., 1998] L. Vercouter, P.Beaune, and Sayettat. Apprentissage dans les sma. In actes des journées française sur l'intelligence artificielle distribuée et les systèmes multi-agents (JFIADSMA'98) Hermès, Paris France, 1998

[Von Neumann et *al*, 1944] J. Von Neumann, et Morgenstern O. Theory of games and economic behavior. Princeton University Press .

[Watkins, 1989] Christopher J. C. H. Watkins. Learning from Delayed Rewards. PhD thesis, King's College, Cambridge, UK, 1989.

[Watkins et al, 1992] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. Machine Learning, 8(3):279-292, 1992.

[Weiss, 99] G. Weiss. “Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence”. , MIT Press, 1999

[Wooldridge et Jenning, 1995] M. Wooldridge, et N. R. Jennings, Intelligent agents : Theory and practice. The knowledge Engineering Review, 10(2): 115-152, 1995

[Wooldridge, 2001] M. Woolridge. Introduction to Multiagent Systems. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[Wooldridge,.2002]. Wooldridge, M. An Introduction to Multi Agent Systems John Wiley and Sons, 2002.

[Zhang et al, 1995] W. Zhang and T. Dietterich. A reinforcement learning approach to jobshop scheduling. In IJCAI' 95, 1995.

Autres référence en ligne de l'apprentissage par renforcement (Reinforcement Learning):

Neuro-Dynamic Programming, Bertsekas et Tsitsiklis, 1996.
http://web.mit.edu/jnt/www/ndp.html

Un repository Apprentissage par renforcement :
http://www-anw.cs.umass.edu/rlr/

Reinforcement Learning: A Survey (Kaelbling, Littman and Moore)
http://www.ai.mit.edu/people/lpk/papers/rl-survey.ps

Reinforcement Learning and artificial Intelligence
http://rlai.cs.ualberta.ca/RLAI/rlai.html

Référence en neurosciences : Chapitre 9 du livre de Dayan et Abbott, Theoretical Neuroscience (disponible à la bibliothèque de l'ENS Cachan), article de Doya:
http://www.cns.atr.jp/~doya/papers/index.html

Analyse d'algorithme d'apprentissage par renforcement :
Itération sur les politiques avec approximation linéaire
http://www.cs.duke.edu/~parr/cs-2001-05.ps.gz

Itération sur les valeurs avec approximation :
http://hal.inria.fr/inria-00116987/en/

Itération sur les politiques avec approximation :
http://www.cmap.polytechnique.fr/~munos/papers/jedai.ps.gz

Autres approches :
Gradient de la mesure de performance par rapport à des paramètres de contrôle :
http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume15/baxter01a.pdf

Annexes

Algorithmes d'apprentissage par renforcement

Nous avons dressé le tableau récapitulatif des principaux algorithmes d'apprentissage par renforcement [Sutton et al. 1998] :

Algorithmes	Propriétés	Limites
Critique /Acteur, A.G. Barto en 1983.	- Basés sur une structure de mémoire séparée pour la représentation explicite de la politique de choix d'action ('acteur') et la fonction valeur-état ('Critique'). - Algorithme de contrôle de type "On-policy" - Réactualisation selon la méthode TD. - Choix d'action nécessitant peu de calculs.	✓ Modèle de base important conduisant à des réalisations diverses aux propriétés encore mal connues.
AHC-Learning, R. Sutton en 1984.	Basé sur des fonctions d'évaluation, et de sélection séparées	✓ Problème de conception des deux fonctions. ✓ Propriétés peu maîtrisées (convergence, sensibilité aux paramètres)
Arp, A. G. Barto et P. Anandan en 1985.	- Basé sur un modèle de neurones stochastiques calculant une probabilité d'exécuter une action pour un état donné. - Ne calcule pas de valeur d'utilité. - Adaptable à l'apprentissage de réseau de neurones multicouche avec neurones de sortie binaires et stochastiques. - Convergence prouvée pour une action binaire et entrées linéairement indépendantes.	✓ Actions binaires. ✓ Ne calcule pas de valeur d'utilité.
Q-Learning, Watkins en 1989.	- Basé sur le calcul de la fonction valeur-état-action Q(s,a) - Algorithme de contrôle de type "off-policy" - Basé sur itération de politique généralisé (GPI). - La fonction Q approche directement la fonction optimale Q* - Convergence prouvée sous conditions.	✓ Taille de la mémoire Q(s,a) augmente proportionnellement à |S|*|A| ✓ Problème de généralisation

QCON et AHCON, Lin en 1992	- Algorithmes implémentés sur des réseaux neurones multicouches à une sortie. - Traite le problème de la généralisation - Chaque action possible est associée à un réseau de neurones multicouche.	✓ Celles des méthodes parentes (mais généralisation amélioré) ✓ Problème de taille de mémoire (pour chaque action possible il faut un réseau de neurones). ✓ Pas de généralisation sur les actions
Q-récurrent, Lin en 1992	- Algorithme Q-learning implémenté sur des réseaux neurones récurrents. d'Elman. - Permet de stocker dans les couches internes des réseaux récurrents des informations pertinentes sur le passé du processus.	✓ Celles du Q-leaming (mais généralisation amélioré) ✓ Apprentissage des réseaux récurrents à maîtriser.
Q-RBF, par Andersen en 1993	- Implémentation du Q-leaming sur des réseaux de neurones à fonctions de base radiales (radial basis function (RBF)). - Proposé pour résoudre le problème de la généralisation.	✓ Celles du Q-leaming (mais généralisation amélioré) Estimation du nombre et de la position des gaussiennes nécessaires pour couvrir l'espace de situation.
R-Leaming, Schwartz en 1993	- Pour processus non épisodiques, sans pondération γ sur le renforcement. - Cherche à maximiser l'espérance des gains sur un pas de temps. - Algorithme de contrôle de type "off-po- licy", variante du Q-learning.	✓ Celles du Q-learning ✓ Suppose que tout état est atteignable selon la politique π. ✓ Méthode encore peu expérimentée et aux propriétés mal connues.
B-Learning, T. Langlois en 1993	- Architecture du AHC-learning. - Prise en compte de la dérivé de la fonction valeur état pour la sélection d'action.	✓ Celles du AHC learning
Q-leaming hiérarchique (HQL), F. Kirchner en 1995	- Décompose l'apprentissage d'une tâche en plusieurs apprentissages de tâches plus simples. - Les objectifs définis à un niveau « n » sont atteints si les actions exécutées au niveau « n-1 » sont correctes. - Les renforcements sont attribués de niveau en niveau du haut vers le bas.	✓ Décomposition en sous tâches pas évidente
Q-Kohonen, S.Sehad en 1996	- Algorithme Q-learning avec carte auto-organisatrice de Kohonen. - Un même noeud de la carte code, l'état, l'action et la valeur Q(s,a) associée. - Traite le problème de généralisation (espace d'état et d'action) - Bonnes performances (rapidité, généralisation)	✓ Deux techniques à maîtriser (apprentissage non supervisé (cartes de Kohonen), et apprentissage par renforcement. ✓ Une même carte code les espaces d'état et d'action.

SARSA, Sutton en 1998.	- Basé sur le calcul de la fonction valeur-état-action Q(s,a) - Algorithme de contrôle de type "On-policy". - Actualisation de Q lors du passage d'une paire (s,a) vers un autre paire (s',a'). - Convergence, non prouvée en général, dépend du choix de la stratégie d'exploration	✓ Taille de la mémoire Q(s,a) augmente proportionnellement à |S|*|A| ✓ Problème de généralisation ✓ Sensible à la stratégie d'exploration.
TD(λ),Q(λ), SARSA(λ) R. Sutton en 1998	- Les estimations moyennes calculées avec les renforcements obtenus sur n pas (TD(0) tient compte uniquement du renforcement immédiat) - L'actualisation des fonctions valeurs est propagées aux états et action précédents avec un taux de diffusion λ. - Permet d'accélérer l'apprentissage par rapport aux méthodes TD simples. - Méthodes plus générales et apprentissage plus efficace. - Convergence du TD(λ) prouvée dans le cas linéaire par P. Dayan.	✓ Celles des méthodes parentes (TD.SARSA...)
ρ-learning, J.M. Porta en 2000	- Inspiré du Q-leaming, permet de sélectionner automatiquement le groupe de capteurs fournissant les informations les plus pertinentes pour l'apprentissage - Q-learning avec plusieurs tables de fonction valeurs mises à jour, chacune étant liée à un groupe de capteurs - L'apprentissage avec ρ-leaming est plus rapide que Q-leaming.	✓ Augmentation du temps de calcul avec le nombre de groupes de capteurs retenus. ✓ Convergence de l'algorithme non prouvée ✓ Problème de généralisation non résolu
Apprentissage par renforcement des règles, M. Svinin en 2001	- Apprentissage des règles les plus pertinentes pour atteindre un objectif. - Le système de décision est conçu selon le choix des règles en compétition, la sélection d'une action, le calcul de l'utilité d'une règle et génération d'une nouvelle règle.	✓ Peu expérimenté, propriétés mal connues.

L'environnement de simulation basé agent Netlogo

1 Introduction

La simulation basée agent fait, depuis longtemps, partie des moyens que la science s'est donnée pour saisir et représenter des phénomènes dynamiques complexes étudiés dans tous les domaines de recherche. C'est particulièrement en sciences humaines que ce type d'approche se montre particulièrement riche, permettant d'explorer d'une façon rigoureuse les liens existant entre les diverses échelles du phénomène social ; d'aller, par exemple, au bout de nombreuses hypothèses portant sur le lien entre des stratégies de comportements individuelles et de phénomène social global qui en découle.

Parmi les nombreux logiciels permettant de faire usage de tels modèles se distingue notamment « NetLogo ». C'est un environnement fort intéressant pour simuler les phénomènes dynamiques complexes mis à disposition par le Center for Connected Learning and Computer-Based Modelling de la Northwestern University.

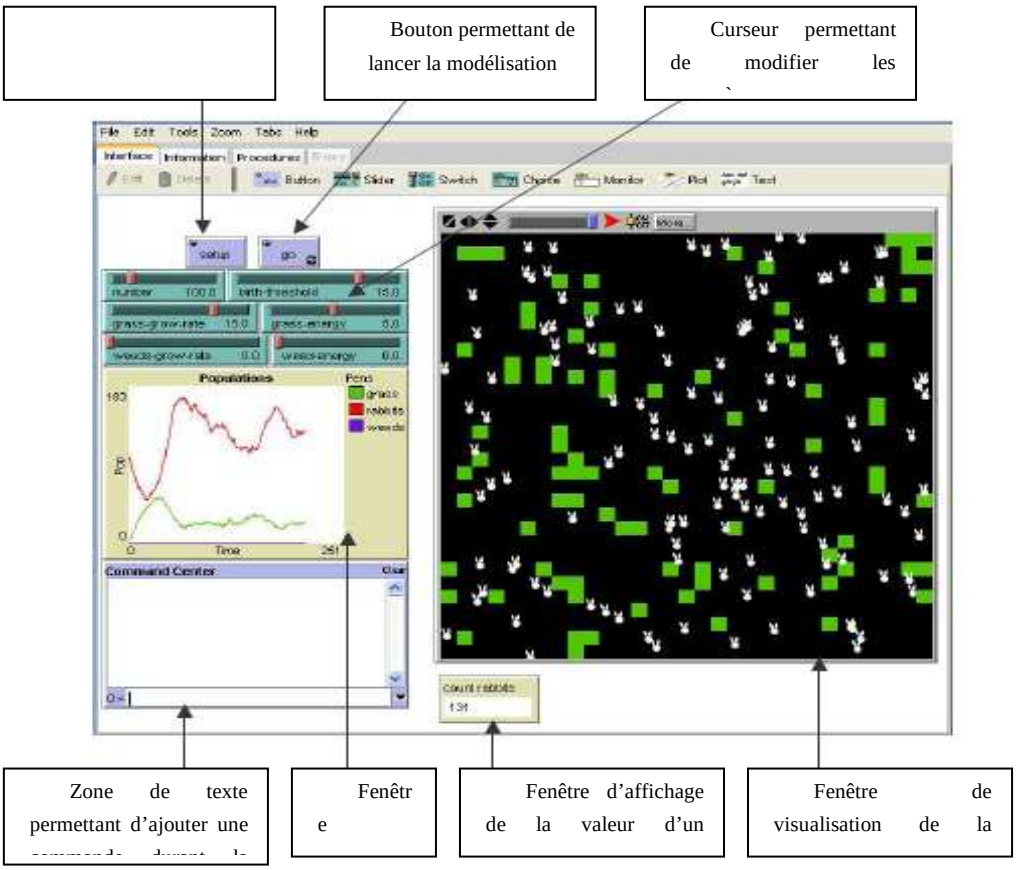
C'est aussi un logiciel libre Basé *Java*, qui peut être installé sur toutes les plateformes communes (*Linux*, *Windows*, *Mac OS*). Le langage d'instructions permettant de définir les comportements des agents simulés est particulièrement facile à prendre en main.

2 Présentation de NetLogo

2.1 Description

La version 3.1.4 de *NetLogo,* offre entre autres un affichage en trois dimensions permettant de mieux distinguer les agents mobiles des cellules du maillage spatial sur lequel ils se déplacent et avec lequel ils interagissent.

2.2 L'interface graphique : l'interface graphique de NetLogo se présente comme suit :



Le code de la simulation est accessible dans l'onglet « procédures ». Les boutons, les curseurs, les fenêtres d'affichage de courbes et les fenêtres d'affichage de paramètres sont créés à partir de la barre d'outils de l'interface graphique.

3.1 Les boutons : Ils fournissent un moyen simple de contrôler le modèle.

En général un modèle a :

– Un bouton «**setup**» pour initialiser l'état du monde
– Un bouton «**go**»pour exécuter le modèle en continue
– Éventuellement d'autres boutons pour exécuter d'autres actions

Un bouton contient du code Netlogo qui s'exécute quand on clique sur le bouton

–«**once-button**» : le code est exécuté une fois
–«**forever-button**» : le code s'exécute indéfiniment jusqu'à ce qu'on le clique à nouveau, ou si le code exécute la commande «stop»

Il faut spécifier qui exécutera le code associé au bouton :

- ✓ l'observateur
- ✓ Toutes les tortues
- ✓ Tous les patches
- ✓ autres cas particuliers

2.3 La gestion des fenêtres d'affichage :

En ce qui concerne l'affichage d'une courbe : après avoir créer une fenêtre permettant

l’affichage d’une courbe dans l’interface graphique, il suffit de mettre à jour à chaque pas de temps la valeur de la courbe (ou des courbes) que l’on souhaite afficher.

Considérons les lignes de code suivantes :

```
set-current-plot "population"
set-current-plot-pen "total"
plot count turtles
```

La fonction set-current-plot permet de spécifier que l’on va mettre à jour la fenêtre d’affichage appelée « population ». On spécifie alors que l’on va considérer la courbe appelée « total » de cette fenêtre à l’aide de la fonction set-current-plot-pen. Finalement on met à jour la valeur de la courbe en fournissant en paramètre de la fonction plot l’opération à effectuer.

En ce qui concerne l’affichage d’une fenêtre contenant un paramètre (qui est une variable globale) : après avoir créer une fenêtre permettant l’affichage d’un paramètre, il suffit de remettre à jour à chaque pas de temps la valeur de ce paramètre.

3 Eléments de programmation

3.1 Les Acteurs principaux

NetLogo permet de simuler les interactions entre un grand nombre d'agents ou agent et environnement, puis de voir ce qui se passe lorsqu’un agent est programmé pour suivre des règles spécifiques. Les agents vivent sur un support 2D constitué de cellules appelées « patches ». Ces cellules ont des coordonnées.

Un agent peut être une personne, un animal, un insecte, une cellule, etc... Cet agent a la particularité de pouvoir suivre des instructions ou des lois. En effet, son comportement est déterminé par un ensemble de lois, en général des lois stimulus-réponse typiques et simples.

Il y a trois types d'agents : Les tortues (turtles). Les patches, et l'observateur.

- ✓ **Les tortues** : dans NetLogo, on peut manipuler des entités, qu'on peut appeler fourmis, termites, loups, personne, etc. Ce sont des sortes de races. On fait remarquer que par défaut ces entités sont dénommées tortues. Ils sont appelés « tortues » en hommage au langage de programmation dont dérive NetLogo dans lequel le programmeur contrôle des tortues sur un écran. Les tortues répondent à des commandes ou des fonctions. Elles peuvent être des plusieurs sur l'écran au même moment. Les tortues ont pour coordonnées xcor et ycor.
- ✓ **Les patchs** : Les patchs constituent l'environnement des tortues, ce sont les cases de l'écran principal de NetLogo. Grâce à la programmation des patchs, on peut donc faire intervenir l'environnement des tortues; ceci permet d'établir des modèles beaucoup plus proches de la réalité, quand on sait l'importance des interactions entre les individus et leurs environnements dans l'établissement des comportements. Chaque patch est un carré (une case) sur lequel les tortues peuvent se déplacer. Le patch central a pour coordonnées (0,0). Les coordonnées d'un patch sont appelées pxcor et pycor. Le nombre total de patches est déterminé par les variables max-pxcor et max-pycor.

Les coordonnées des patches sont des entiers. Notons que le support des patches n'est pas borné. Chaque patch a le même nombre de voisins. Ainsi si une tortue disparaît d'un côté de l'écran, elle réapparaît du côté opposé.

- ✓ **L'observateur :** C'est une sorte de superviseur dans le monde des tortues. L'observateur est extérieur au monde dans lequel vivent les tortues mais il peut superviser et ordonner aux tortues de faire certaines choses. C'est lui que l'on utilise lorsque l'on veut gérer un programme dans NetLogo

II permet également d'intervenir alors même que le programme est déjà lancé. L'observateur peut être utilisé pour attribuer des ordres spécifiques à des patches ou à des tortues. Il collecte également des données pour créer des graphiques.

3.2 Les commandes Elles définissent les actions à exécuter par les «agents»
Elles sont de 2 types :

- ✓ **Primitives**: commandes pré-définis dans NetLogo
- ✓ **Procédures**: commandes définis par le programmeur

Une commande **procédure** est définie par un **nom**, précédé par le mot clé **to**, et se terminant par le mot clé **end**
Exemple : commande **procédure setup**

```
To setup
    ca      ;; nettoyer l'écran
    crt 10      ;; créer 10 tortues
end
```

Pour les Commandes primitives, ce sont des commandes prédéfinies dans NetLogo

Exemples :

–ca (clearall),
–crt (create turtles),
–lt (left turn),
–rt(right turn),
–fd (forward),

et peuvent avoir des valeurs en entrée :

– crt100
– rt50
–…

3.3 Reporters : Les Reporters exécutent une opération, et retournent un résultat soit à une commande soit à un autre reporter (un peu comme une fonction). Il existe deux types de reporters :

- Reporters primitives : prédéfinis dans NetLogo Exemple : la fonction Random.
- Reporters procédures : définis par le programmeur.

3.4 Définition des variables :

Les différentes variables se définissent de la manière suivante:

- Variable globale : globals [clock]
- Variable tortue : turtles-own [energy speed]

On utilise la commande “set” pour les initialiser. Par défaut, elles sont initialisées à 0. Les variables globales peuvent être lues et modifiées par n’importe quel agent à tout moment. Une tortue peut lire et modifier les variables patches du patch sur lequel elle agit. Lorsque l’on veut qu’un agent lise ou modifie la variable d’un autre agent, il faut mettre –of après le nom de la variable et spécifier, après, de quel agent il s’agit.

Exemples:

Set color-of turtle 5 red ; la tortue n° 5 devient rouge.

C’est une variable exclusive à une fonction.

On la définit de la manière suivante : locals [var1 var2 …]

Ces variables ne sont utilisables qu’à l’intérieur de la fonction. Elles doivent être définies au début de la fonction, avant toute commande.

3.5 Les variables prédéfinies :

Il existe plusieurs variables prédéfinies relatives aux variables « tortues » :

breed : (= race)

Elle permet de sélectionner l’ensemble des tortues de la même race.

breeds:

Ce mot clé permet de définir l’ensemble des races. Il ne peut être utilisé qu’en début de programme, avant toute définition de fonction.

color : Elle définit la couleur d’une tortue.

heading : Elle indique vers quelle direction la tortue est dirigée.

heading $\in$ [0 ; 360].

0 = Nord, 90 = Est, 180 = Sud, 270 = Ouest

label : Elle permet d’associer une valeur à une tortue. Cette valeur peut être de n’importe quel type.

label-color : Elle détermine la couleur du label d’une tortue (si elle en a un).

label-color $\in$ [0 ; 140].

size : La taille par défaut est 1.0. La tortue a la même taille qu’un patch. Toutes les tortues ont la même taille sauf si on coche la case « turtle-sizes ».

xcor, ycor : Ce sont les coordonnées de la tortue.

Il existe également plusieurs variables patches prédéfinies : pcolor, plabel, plabel-color, pxcor, pycor. Leur fonction est la même que pour les variables tortues.

3.6 Les fonctions : Une fonction se définit de la manière suivante :

to nom_fonction corps_de_la_fonction end

Les fonctions prédéfinies du langage NetLogo sont répertoriées dans une page du logiciel appelée The Primitives Dictionary. Il y a deux fonctions à définir obligatoirement dans un programme NetLogo : la fonction setup et la fonction go. C'est à l'utilisateur de définir ces deux fonctions. La fonction setup définit l'état initial du modèle. La fonction go démarre le processus de modélisation.

3.7 Les structures

La structure if : Syntaxe : if condition [traitement]

Exemple : ask patches [if pxcor > 0 [set pcolor blue]

La structure ifelse:

Syntaxe: ifelse condition [traitement_si_la_condition_est_vérifiée] [traitement_sinon]

Exemple : ask patches [ifelse pxcor > 0 [set pcolor blue] [set pcolor red]]

La structure ask:

Cette structure permet d'appliquer une ou plusieurs instructions à toutes les variables d'un même type. En effet, la structure suivante :

ask turtles

[

…..]

permet d'appliquer à toutes les variables de type tortue les instructions qui se situeront dans le

corps de la structure.

Exemple de Netlogo

```
To setup                    ; Procédure setup
Ca
Crt 100
Ask turtles
[set color red          ; colorer les tortues en rouge
rt random 360           ; orienter chacune au hasard fd 50 ]; les faire avancer de 50 pas
Ask patches
    [if (pxcor > 0)             ; colorer les patches du coté droit
    [set pcolor green] ]        ; de l'écran en vert
End
```