Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs

THÈSE

présentée et soutenue publiquement le 10 septembre 2003

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1
(spécialité informatique)

par

Olivier BUFFET

Composition du jury

Président :	René Schott	Professeur à l'Université Henri Poincaré - Nancy 1
Rapporteurs :	Marie-Odile Cordier	Professeur à l'Université de Rennes I
	Michael L. Littman	Associate Research Professor, Rutgers University
Examinateurs :	Alain Dutech	Chargé de recherche, INRIA
	Jean-Arcady Meyer	Directeur de recherche, CNRS
Directeur de thèse :	François Charpillet	Directeur de recherche, INRIA

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Résumé

Apprentissage par renforcement (A/R) et systèmes multi-agents (SMA) sont des outils prometteurs dans le domaine de l'intelligence artificielle : le premier permet de concevoir le comportement d'entités intelligentes (agents) à l'aide de simples récompenses (on se place ici dans le cadre des processus de décision markoviens), et le second se fonde sur l'idée qu'un comportement intelligent peut "émerger" de la collaboration d'un groupe d'agents.

Nous avons cherché, dans cette thèse, à explorer les moyens d'utiliser conjointement ces deux outils. Les deux principales parties de ces travaux présentent des points de vue symétriques : 1- comment concevoir des agents coopérants par des méthodes d'apprentissage par renforcement, et 2- comment améliorer un agent (apprenant par renforcement) en lui donnant une architecture interne distribuée (de type SMA). Ces aspects ont tous deux amené à des approches incrémentales, comme détaillé ci-après.

1. Dans le cadre Multi-Agents, nous nous attaquons au problème de la conception d'agents réactifs collaborant à l'aide d'approches par A/R. Comme divers problèmes théoriques difficiles apparaissent, nous avons proposé d'utiliser des méthodes progressives (un apprentissage progressif commençant avec des situations simples) de manière à aider les agents dans l'apprentissage de leurs comportements.

   Les expérimentations montrent que certains optima locaux des algorithmes classiques d'A/R en-ligne peuvent être dépassés à travers cette méthode. Malheureusement, les algorithmes progressifs sont difficiles à automatiser dans les cadres partiellement observables, ce qui en limite l'usage.
2. Dans le second cadre, nous avons travaillé sur la décomposition d'une politique en plusieurs politiques parallèles au sein d'un agent. Un tel problème entre dans le champ de la sélection d'action, qui s'intéresse à la prise de décision quand différents objectifs simultanés sont pris en compte.

   Une première étape a été de proposer et d'étudier des algorithmes pour combiner des politiques de manière adaptative. Les résultats étaient encourageants, puisque certains optima locaux ont, ici aussi, été dépassés : les techniques d'A/R classiques se limitent habituellement à seulement éviter des problèmes immédiats.

   Ensuite, un algorithme novateur a été proposé pour que l'agent trouve de manière automatique les politiques de base qu'il requière dans un environnement donné. A notre connaissance, c'est le premier travail dans lequel l'agent trouve et apprend de manière autonome les "comportements de base" nécessaires à sa prise de décision : d'habitude, ceux-ci sont fournis par un concepteur humain.

En résumé, les deux parties de cette thèse de doctorat rassemblent l'idée des approches par décomposition (au travers d'un SMA ou d'une architecture de sélection d'action) et l'apprentissage par renforcement, avec dans les deux cas une conception progressive des agents. Du fait d'hypothèses difficiles (observations partielles, pas de modèle...), les deux parties du travail présenté sont basées sur des heuristiques, ce qui ne les empêche pas d'être très prometteuses pour la conception d'agents.

Mots-clés: apprentissage par renforcement, systèmes multi-agents, processus de décision markoviens, sélection d'action, apprentissage progressif

Abstract

Reinforcement Learning (RL) and Multi-Agent Systems (MAS) are promising tools in the field of Artificial Intelligence : the former allows the design of behaviors for smart entities (agents) thanks to simple rewards (we are here in the framework of Markov Decision Processes), and the later is based on the idea that a smart behavior may "emerge" from the collaboration of a group of agents.

We have tried, in this thesis, to explore the ways of conjointly using both tools. The two main parts of this work present symetric points of view : 1- how to design cooperating agents through reinforcement learning methods, and 2- how to improve a (reinforcement learning) agent with a distributed internal architecture (such as a MAS). These aspects have both led to incremental approaches, as detailed below.

1. In the Multi-Agent framework, we address the problem of conceiving collaborating reactive agents through RL approaches. As various difficult theoretic problems arise, we have proposed to use shaping methods (a progressive learning beginning with simple situations) in a view to help agents learn their behaviors.

   The experiments show that some local optima of classical on-line RL algorithm can be overcome through this method. Unfortunately, shaping algorithms are difficult to automate in partially observable frameworks, what limits their use.

2. In the second framework, we have worked on the decomposition of a policy into various parallel policies in an agent. Such a problem lies in the field of Action-Selection, which concerns making a decision when considering different simultaneous goals.

   A first step was to propose and study algorithms to adaptively combine policies. The results were encouraging, as some local optima have also been overcome here : classical RL techniques are usually limited to only avoiding immediate problems.

   Then, an innovative algorithm has been proposed for the agent to automatically find the basic policies it requires in a given environment. To our knowledge, this is the first work where the agent autonomously finds and learns the “basic behaviors” necessary to its decision-making : these are usually given by a human designer.

To sum up, both parts of this PhD thesis bring together the idea of decomposition approaches (through a MAS or an Action-Selection architecture) and Reinforcement Learning, with a progressive building of the agents in both cases. Due to difficult hypotheses (partial observations, no model...), both parts of the presented work are based on heuristics, what does not prevent them from being very promising to design agents.

Keywords: reinforcement learning, multi-agent systems, Markov decision processes, action selection, shaping

Remerciements

« Lao Tseu a dit : «il faut trouver la voie». Donc, je vais vous couper la tête. »

(Didi, fils de Wang Jen-Ghié dans) Le Lotus Bleu, Hergé

Voilà. Désolé d'avoir été si long (pour ceux qui s'essayent à la lecture de ce document). Ce n'est pourtant pas l'habitude de l'auteur d'être si prolixe. En plus il n'a pas un style littéraire des plus légers : c'est pas de chance ça...

Mais bon, c'est fait maintenant, Olivier il est docteur. Ca n'a pas été sans mal : il a fallu tout plein de gens pour en arriver là. Je vais donc essayer maintenant de remercier comme il se doit les personnes ayant contribué d'une manière ou d'une autre à ce qu'on en arrive là.

Respectant ce qui semble être une tradition, je commencerai par le jury en remerciant Marie-Odile Cordier, Michael Littman (auquel j'espère encore pouvoir rendre visite), Jean-Arcady Meyer et René Schott. Ces remerciements vont aussi à Claude Kirchner, qui a suivi mon travail en tant que référent pendant ces trois années.

Une autre assemblée que je me dois de remercier est celle qui a participé à mon comité de thèse : Frédéric Alexandre, Frédérick Garcia, Manuel Samuelides et Olivier Sigaud. Les quatre heures de discussion furent difficiles sur le moment, mais ce travail constructif fut largement récompensé. Je ne peux qu'encourager mes camarades à passer par là eux-aussi.

Evidemment, j'ai omis jusqu'ici de mentionner mes deux chefs : François et Alain. J'aurais difficilement pu souhaiter meilleurs relations avec mes encadrants, lesquels ont su m'aider à avancer, m'apportant leurs points de vue complémentaires et me guidant dans ma méthode de travail. J'ai encore du chemin à faire, et espère pouvoir continuer en aussi bonne compagnie.

Plutôt que de compagnie, c'est peut-être de troupe dont je devrais parler pour remercier mes compagnons de fortune, travailleurs parfois fonctionnaires mais souvent précaires, certains déjà partis d'autres arrivant tout juste. Mes remerciements se dirigent naturellement vers l'équipe MAIA, mais aussi vers bien des gens de RFIA, ISA, et CÆTERA : Yot, Yann, Yassine et Salma, Virginie, Vincent, Victor, Simon, Renatoto, Régis, Rédimé, Raghav, Olivier, Nono, Mouna, Loïc, Laurent, Jean-Luc, Iadine, Hubert, Hacène, Grom, Glouk, Franck, Fabrice, Eric, David, Daniel, Dahbia, Cédric, Bruno, Blaise, Armelle, __________[1]...

Pour être juste, je ne dois pas oublier ce que m'ont apporté Marie-Claude Portmann et mes collègues ou élèves des Mines, lesquels m'ont permis de découvrir le côté sombre de l'enseignement. Au LORIA, j'ai aussi une pensée pour mes collègues nyctalopes anonymes des derniers mois de rédaction, et pour quelques personnes efficaces et souriantes : Nadine, Martine, Sabrina, Matthieu et le chef pour citer quelques exemples représentatifs.

Pour finir, mes remerciements vont sortir pour partie du cadre professionnel pour aller : vers mes camarades de Supélec et particulièrement ceux qui m'ont accompagné dans l'aventure de la recherche (Olivier et Vincent), vers mes amis (OlivierP, Philippe, Stéphane et Thomas), vers les chouettes et les écureuils de Villers, ainsi que les belles vosgiennes, les cockers et sharpei, et les ours crétins, tous ayant contribués à leur façon à la chaleureuse atmosphère de ces trois années. Enfin mes derniers remerciements vont vers ma famille, dont la contribution ne date pas d'hier.

[1] Complêter par votre prénom en cas d'oubli.

« Continuer à rêver. »
Un chat idéal, dans "Réunion d'avancement de thèse parfois bimensuelle".

← http ://www.m4de.com

Table des matières

[2] Regarde maman ! Sans les p'tites roues !

Table des figures

Liste des tableaux

Liste d'Algorithmes

Introduction

« L'intelligence, c'est la faculté d'adaptation. »

André Gide

Le manuscrit que vous avez sous les yeux présente le travail qui a été effectué au cours d'une thèse en informatique, plus précisément dans le domaine de l'intelligence artificielle.

Dans cette introduction, nous allons expliquer la démarche suivie dans le présent mémoire. Dans ce but, nous partons du problème que pose la définition de l'intelligence (section 0.1)[3]. De là, nous abordons le domaine de l'intelligence artificielle (IA) elle-même (section 0.2), en mettant en avant des thèmes auxquels nous souhaitons nous intéresser plus particulièrement. Ces thèmes vont permettre d'abord de mettre en avant deux approches de l'IA autour desquelles va se construire notre travail (section 0.3) : apprentissage par renforcement et systèmes multi-agents. Puis sera enfin présenté le problème précis qui va retenir notre attention (section 0.4) : comment ces deux approches peuvent-elles se compléter ? On pourra alors finalement établir le plan selon lequel est organisé ce manuscrit (section 0.5) avant d'y entrer réellement.

1 Intelligence

Savoir ce qu'est l'intelligence est une vaste question. Nous nous contentons ici d'une discussion assez restreinte, mais qui devrait suffir à nos desseins.

1.1 Définition

En consultant l'encyclopédie libre en-ligne Wikipedia[4], on pouvait trouver au mot “intelligence” l'article suivant (qui a pu évoluer depuis) :

> *« Il y a une large controverse quant à ce qu'est exactement l'intelligence. Néanmoins, on peut la définir grossièrement par la capacité à acquérir et appliquer de la connaissance. Ces ingrédients incluent la perception d'un environnement, la capacité de tirer des conclusions sur l'état de cet environnement, et la communication de ces conclusions. Dans la plupart des cas, le concept d'environnement recquiert une entité autre que celle qui possède l'intelligence, mais la connaissance de soi peut être suffisante.*
>
> *Voici une définition plus compréhensible : “capacité à s'adapter effectivement à l'environnement, soit en changeant soi-même, soit en changeant l'environnement, soit en en trouvant un nouveau” (Encyclopaedia Britannica). »*

Cette idée que l'*intelligence* est la capacité à s'adapter à son environnement se retrouve aussi chez Piaget, dont les travaux se sont principalement intéressés au développement de l'intelligence chez l'enfant (on peut citer parmi bien d'autres ouvrages [Piaget, 1967]). Si d'autres points de vue sur l'intelligence existent, nous faisons le choix de garder cette idée comme fil conducteur de notre propos.

[3]Pour des raisons pratiques, ce chapitre d'introduction porte le numéro 0.

[4]`http ://www.wikipedia.org/`

1.2 Adaptation et but

En définissant l'intelligence comme nous l'avons fait, on a en réalité reporté la difficulté sur la notion d'*adaptation*. Pour pallier ce défaut, commençons simplement par regarder une définition assez générale du verbe transitif "adapter" (tirée ici du trésor de la langue française) :

> «*[Le sujet désigne une personne ou une force agissante concrète ou abstraite]* **Adapter quelqu'un ou quelquechose à quelqu'un ou à quelquechose.** *Mettre en accord, approprier à quelque chose ou à quelqu'un d'autre, considéré comme prépondérant ou du moins comme incontestablement réel, de manière à obtenir un ensemble cohérent ou harmonieux.*»

Cette première définition est intéressante, mais difficile à relier au sujet qui nous intéresse. Une autre définition (de l'adaptation cette fois), tirée du domaine de la psychologie, fait clairement le lien entre cette première définition et l'intelligence vue par Piaget (ou définie dans l'Encyclopædia Britannica) :

> «*L'ensemble des activités par lesquelles un individu modifie ses conduites pour s'ajuster de manière optimale à un milieu déterminé.*»

On notera juste que le caractère *optimal* du résultat obtenu peut être discutable. L'adaptation signifie-t-elle que la meilleure solution va être obtenue ?

Nous proposons pour notre part d'élargir cette idée de "s'ajuster" à un milieu pour considérer plutôt que *s'adapter* c'est évoluer de manière à atteindre un but, un objectif, ou à s'en rapprocher. On rejoint d'ailleurs ainsi un autre point de vue pour lequel l'intelligence est la faculté de répondre à des problèmes nouveaux (en mettant "problème" et "but" en correspondance).

Revenons à l'intelligence

Ainsi, considérons une entité en présence d'un objet qu'elle observe, sur lequel elle peut agir et par rapport auquel elle a un but, sans avoir connaissance du moyen d'atteindre ce but (voir figure 1). L'intelligence est le processus d'évolution, d'adaptation de l'entité qui lui permet d'élaborer précisément ce moyen d'atteindre le but : les actions à effectuer face aux diverses observations possibles. L'objet dont il est question ici est ce qui, généralement, est appelé "environnement" : le domaine extérieur avec lequel l'entité est en interaction.

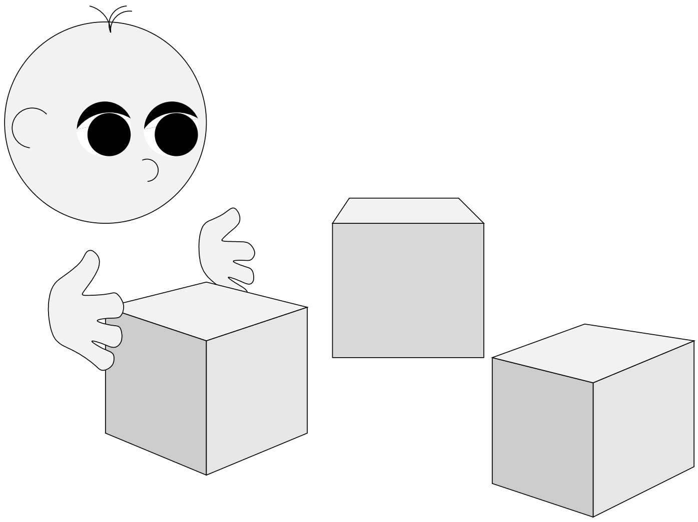


Fig. 1 – Bébé joue avec ses cubes : il les manipule, observe ainsi ce qu'il peut faire avec, et en déduit un moyen d'arriver à ses fins.

Ce processus d'adaptation peut prendre diverses formes :

– Il peut s'agir d'un raisonnement construit sur un ensemble de connaissances a priori et en produisant d'autres.
– L'acquisition de connaissances qui va permettre d'atteindre le but fixé peut aussi passer par un apprentissage par l'expérience (de type essai-erreur par exemple).

Note : Des situations un peu particulières peuvent être recouvertes par l'idée de processus intelligent que nous venons de décrire. Il se peut par exemple que l'entité soit sans action possible et que le but soit de comprendre le fonctionnement de l'objet. L'entité peut même être son propre objet de réflexion, faisant ainsi de l'introspection.

1.3 But, libre arbitre

Hélas, même en éclaircissant l'idée d'adaptation, on peut à nouveau dire qu'une partie du problème n'a encore été que déplacée. Une question qui se pose en effet est de savoir dans quels cas une entité évolue en étant guidée par un but.

On peut par exemple facilement douter qu'il y ait une véritable idée de "but" chez l'ascidie, petit animal marin qui digère son cerveau une fois fixé sur un rocher dont il ne bougera plus. Ou alors, il faut accepter l'idée qu'un objectif peut être ce qu'un système (en fin de compte automatique) tente d'atteindre.

Ce sont à des débats délicats qu'on arrive en s'intéressant à la notion de but. En cherchant à savoir ce que le but d'un système artificiel peut être, et en s'intéressant au cas des systèmes naturels, on risque alors de rejoindre le problème de l'existence du libre arbitre : à savoir si ce but est un réel choix ou si ce n'est que l'observateur qui interprête les choses comme telles.

Les questions ainsi soulevées par la notion d'intelligence deviennent assez difficiles et sortent de notre champ de travail. Nous allons en rester là sur ce sujet (laissant le lecteur à ses réflexions) pour avancer vers les questions abordées dans ce manuscrit, et donc parler d'abord d'intelligence artificielle.

2 Intelligence Artificielle

2.1 Définition

Autant que définir l'intelligence, définir le domaine de l'*intelligence artificielle* (ou IA) prête à discussions, même si l'on s'accorde à dire qu'il s'agit à la base de *concevoir des systèmes intelligents.*

La réponse apportée dans [Russell et Norvig, 1995] est des plus instructives. Elle propose d'organiser les points de vue sur ce qu'est l'IA en quatre catégories, selon les réponses aux deux questions suivantes :

- S'agit-il d'abord de considérer, dans la conception, la manière de *penser* du système (comment le système fonctionne et atteint ses objectifs) ou plutôt la manière d'*agir* (le comportement observé) ?
- S'agit-il d'autre part de *reproduire* ce que l'on appelle l'intelligence (chez l'humain par exemple) ou plutôt de concevoir un système *rationnel* (c'est-à-dire cherchant à atteindre au mieux un objectif donné) ?

Pour illustrer cette catégorisation, nous reproduisons dans la table 1 des exemples de définitions de l'intelligence artificielle rangés selon ces deux caractéristiques.

TAB. 1 – Quelques définitions de l'IA. Elles sont organisées en quatre catégories (tirées de [Russell et Norvig, 1995] page 5) :

Les systèmes qui pensent comme des humains.	Les systèmes qui pensent rationnellement.
Les systèmes qui agissent comme des humains.	Les systèmes qui agissent rationnellement.

"Le nouvel et excitant effort pour faire penser des ordinateurs...*des machines dotés d'esprits*, au sens plein et littéral" [Haugeland, 1985] "[L'automatisation des] activités que nous associons à la pensée humaine, activités telles que la prise de décision, la résolution de problèmes, l'apprentissage..." [Bellman, 1978]	"L'étude des facultés mentales à travers l'utilisation de modèles computationnels" [Charniak et McDermott, 1985] "L'étude des calculs qui rendent possible perception, raisonnement et action" [Winston, 1992]
"L'art de créer des machines assurant des fonctions qui requièrent de l'intelligence quand elles sont assurées par des hommes" [Kurzweil, 1990] "L'étude de la manière de faire faire aux ordinateurs des choses pour lesquelles, jusqu'à maintenant, les hommes sont meilleurs" [Rich et Knight, 1991]	"Un champ d'étude qui cherche à expliquer et émuler le comportement intelligent en termes de processus computationnels" [Schalkoff, 1990] "La branche de l'informatique concernée par l'automatisation du comportement intelligent" [Luger et Stubblefield, 1993]

L'optique dans laquelle nous nous plaçons est celle des "systèmes qui agissent rationnellement". Cela fait d'ailleurs écho à la définition de l'intelligence que nous avons apporté, puisqu'il s'agit de concevoir une entité qui cherche à atteindre un but, laquelle entité sera souvent appelée "agent". Dans ce cadre, l'intelligence artificielle suppose un concepteur qui définisse l'architecture de l'agent (autant interne : algorithme, qu'externe : capteurs et effecteurs), ainsi que l'objectif qu'il suivra.

Note : pour ne faire que brièvement donner une idée du vaste domaine que représente l'intelligence artificielle, ajoutons qu'il se décompose en de nombreuses branches, selon deux dimensions :
- la dimension des *problèmes* : reconnaissance des formes, démonstration de théorèmes, robotique autonome... et
- la dimension des *outils* : réseaux de neurones artificiels et systèmes à base de connaissance par exemple.

2.2 Point de vue adopté

Si nous avons dit voir l'IA comme le problème de **conception** de *systèmes qui agissent rationnellement*, nous pouvons préciser encore le point de vue que nous adoptons dans nos travaux. Celui-ci est dirigé par trois principaux aspects qui ont retenu notre attention, aspects que l'on peut relier à l'intelligence "naturelle" :

1. On cherche à *automatiser* cette conception en se basant sur un principe simple : mettre au point un système *évolutif* guidé par ses buts. Mais cette idée d'automatisation crée en fait une forte redondance, puisqu'elle revient à dire qu'on va concevoir un système qui s'adapte de manière à être intelligent (c'est-à-dire de manière à être adaptatif). Il est en fait préférable d'insister plus simplement sur notre souhait de limiter autant que possible l'intervention du concepteur.
2. Un autre point est que l'univers dans lequel baigne notre système peut être perçu de manière partielle, son fonctionnement être imparfaitement connu et les actions avoir des résultats hasardeux, ce qui le rend incertain. De plus, notre système n'est pas censé avoir de connaissances a priori sur son environnement, donc aucune représentation symbolique pré-définie (les symboles ne peuvent qu'apparaître). Ces deux problèmes, incertitude et ancrage des symboles [Harnad, 1990], rendent les approches de l'IA dite symbolique peu adéquates. On va donc se placer dans le cadre de l'IA numérique et plus précisément de l'IA située.

   On peut reprendre ici l'exemple de la figure 1, lequel présente un bébé apprenant à manipuler ses cubes. L'incertitude est ici due à différents facteurs : bébé ne connait pas bien les lois de la physique des cubes (problème lié à la modélisation), perçoit mal les formes et n'a qu'une vision très locale (pas de mémorisation de ce qui a été vu), et peut s'avérer assez maladroit (difficultés de préhension). Ce sont autant de raisons qui rendent délicat l'emploi de symboles, sans oublier qu'au départ bébé ne connaît pas les concepts de cube, de chute d'un objet...
3. Enfin, le système considéré peut être une entité seule aussi bien qu'une entité parmi d'autres, ou même être un groupe d'entités. Le groupement d'agents est non seulement une situation dont il faut tenir compte, mais peut être un outil pour s'attaquer à certaines situations. Ceci va nous amener à mettre en avant chez une entité intelligente ses capacités de coopération, d'adaptation à la présence d'autres entités aussi intelligentes, et d'effleurer même la notion d'intelligence sociale (l'idée aussi présente chez [Piaget, 1967] qu'une société peut être intelligente, s'adaptant génération après génération à son environnement).

   L'exemple du bébé reste ici intéressant, puisque son développement est largement guidé par son interaction avec les humains qui composent son entourage.

Ces trois aspects recoupent trois points de vue majeurs sur la notion d'agent : l'agent rationnel de Russell et Norvig, l'agent situé/incorporé de Brooks, et l'agent social auquel s'intéresse par exemple Ferber. Mais nous y reviendrons plus précisément dans les premiers chapitres de cette thèse, sachant que tous trois resteront d'à peu près égales importances dans les travaux qui vont être présentés.

3 Deux approches

A partir des trois aspects qui viennent d'être évoqués (conception automatique, IA située et intelligence sociale), on va faire le lien vers deux branches parmi les outils de l'IA, branches sur lesquelles va reposer cette thèse : l'apprentissage par renforcement d'une part, et les systèmes multi-agents d'autre part.

3.1 Expérimentation

L'exemple donné de comportement intelligent (bébé) met en valeur l'idée que l'apprentissage par l'expérimentation est une étape essentielle. Dénuée de connaissances au départ, l'entité intelligente doit observer son environnement, interagir avec lui de manière à en tirer des informations quant à son fonctionnement. Bébé fait cet apprentissage par l'expérimentation en constatant qu'un cube ne tient pas en l'air tout seul, qu'il peut déplacer un cube, le tourner...

Revenant à l'idée que le système est guidé par un but, apprendre par l'expérience le comportement qui permet d'atteindre ce but (s'il ne s'agit pas simplement de comprendre le fonctionnement de l'environnement) revient à faire un apprentissage par essai-erreur plus ou moins élaboré. Si bébé veut faire une pile de cubes, il peut ainsi essayer les différentes manipulations possibles et ne retenir que celles qui aboutissent[5].

Une telle forme d'apprentissage de comportement par essai-erreur existe en intelligence artificielle, et correspond au domaine de l'*apprentissage par renforcement* (A/R). C'est l'une des deux branches auxquelles nous allons nous s'intéresser. Nous allons même plus précisément travailler dans le champ des processus de décision markoviens, formalisme qui permet de rester dans le cadre de l'IA numérique et, entre autres, de prendre en compte l'incertitude de l'environnement.

3.2 Emergence

Une autre idée qui a été fructueuse en IA est de s'intéresser aux résultats complexes qui peuvent être obtenus à partir de l'interaction d'entités relativement simples.

On peut citer comme phénomènes de ce type le "jeu de la vie" de John Conway [Gardner, 1970], la croissance de plantes ou d'animaux (basée sur des cellules relativement rudimentaires), ou encore le fonctionnement du cerveau (réseau de neurones interconnectés).

Dans chacun des cas sus-cités, chaque entité du groupe obéit à des règles relativement simples qui permettent de définir le comportement de chacune par rapport aux autres, d'organiser le groupe, de transmettre un signal... Le couplage de systèmes dynamiques simples aboutit ainsi à un unique système dynamique complexe.

C'est sur la base de telles idées que se sont développés des travaux sur la thématique des *systèmes multi-agents* (SMA). Ce domaine permet non seulement de mettre en œuvre des systèmes adaptatifs et robustes, mais aussi d'étudier des phénomènes sociaux (collaboration...). Remarquons aussi que l'utilisation de systèmes multi-agents peut être aussi bien une nécessité qu'un moyen :

- On peut, par nécessité, devoir prendre en compte la présence de plusieurs agents qui doivent coopérer (à supposer que les buts ne soient pas conflictuels).

[5]Nous ne suggérons pas pour autant que bébé procède réellement ainsi.

– Mais la “division” d'un système intelligent en plusieurs entités peut aussi être un moyen de faciliter la conception/l'évolution du système en “cassant” la complexité du problème.

4 Mais où est le problème ?

On rencontre malheureusement des difficultés dans chacun des deux domaines que nous venons d'introduire (A/R et SMA). Nous ne citerons que les deux suivantes :

– *SMA :* Si l'on sait pouvoir résoudre efficacement certains problèmes à l'aide de systèmes multi-agents, on manque par contre d'outils pour concevoir de tels systèmes, et plus précisément pour définir le comportement que devra adopter chacun des agents pour que le groupe atteigne l'objectif souhaité.
– *A/R :* L'une des difficultés de l'apprentissage par renforcement (il en est d'autres) est de devoir prendre en compte des situations complexes, dans lesquelles il faut par exemple gérer divers objets tout en poursuivant plusieurs buts.

Considérant donc ces deux difficultés, nous proposons dans cette thèse de voir comment chacun de ces outils de l'IA peut être aidé par l'autre, adoptant ainsi les points de vue symétriques qui suivent :

– d'une part utiliser l'A/R pour concevoir les agents d'un SMA, et
– d'autre part concevoir sous forme d'un SMA une entité basée sur l'A/R.

Dans les deux cas, on verra les approches existantes et ce qu'elles permettent. Dans les deux cas toujours, on proposera des méthodes de conception automatiques et incrémentales, lesquelles seront validées expérimentalement (les résultats théoriques restant limités).

5 Plan

Le présent mémoire s'articule en trois parties principales, la première étant purement bibliographique (pour présenter les outils employés par la suite), et les deux suivantes présentant des études (premier chapitre de chaque partie) et nos apports (chapitres suivants) dans les cadres *A/R pour SMA* et *SMA pour A/R* décrits ci-avant.

I Approches

La bibliographie développée dans la première partie du manuscrit fait le choix de se restreindre aux aspects qui nous seront utiles par la suite, tout en précisant le point de vue que nous adoptons dans notre travail. Le chapitre 1 discutera de la notion d'agent en intelligence artificielle, et présentera le domaine des systèmes multi-agents. Le chapitre 2 reviendra pour sa part sur l'apprentissage par renforcement tel que nous allons en faire usage (sans modèle ni mémoire, sans la propriété de Markov...).

II Apprentissage progressif dans un cadre multi-agents

Comme on l'a dit, la deuxième partie s'intéressera à l'emploi de l'outil “apprentissage par renforcement” pour concevoir des systèmes multi-agents (ou plus précisément les agents qui le forment). Pour commencer, le chapitre 3 fera un état de l'art des travaux du domaine, de manière à mettre en valeurs les problèmes soulevés par cette approche. Proposant dans ce cadre d'*aider* l'apprentissage par des méthodes dites progressives, le chapitre 4 présentera le champ de l'apprentissage progressif, avant de montrer la mise en œuvre qui en a été faite dans nos travaux.

III Combinaison de comportements

La troisième partie, pour sa part, présentera en premier lieu (chapitre 5) une bibliographie sur les méthodes s'attaquant à l'apprentissage par renforcement par des approches SMA. Le chapitre 6 sera consacré à une méthode de sélection d'action de notre cru (une combinaison de comportements) qui s'adapte par renforcement. Le chapitre 7 montrera comment cette méthode permet d'apprendre des comportements en évitant des optima locaux. Et finalement le chapitre 8 bouclera la boucle en décrivant un algorithme qui construit les comportements de base utilisés dans le chapitre 6 à travers le développement progressif d'un arbre par la méthode du chapitre 7.

Un dernier chapitre concluera cette thèse, tirant un bilan des travaux qui y sont présentés et dévoilant les perspectives qui s'y dessinent.

Première partie

Deux domaines

Introduction

« Avec un but dans la vie, on se sent grand, comme si on mesurait quinze centimètres de haut. »
Le Grand Livre des Gnomes (Tome 2 : Les Terrassiers), Terry Pratchett

Les pages qui précédent nous ont permis de présenter le point de vue que nous allons adopter sur l'intelligence artificielle comme domaine visant à concevoir des systèmes adaptatifs, systèmes que nous appellerons "agents". De plus, nous avons insisté sur deux aspects qui vont particulièrement retenir notre attention : l'univers incertain dans lequel baigne l'agent, et le caractère social que peut revêtir l'agent confronté à ses alter ego.

De là est venue l'idée de mettre en présence deux champs de l'intelligence artificielle : les systèmes multi-agents d'une part et l'apprentissage par renforcement d'autre part. Cette première partie du mémoire a pour but de présenter ces deux approches au cours de deux chapitres distincts[6].

Le premier chapitre, qui suit, sera l'occasion non seulement de faire une introduction aux systèmes multi-agents, mais aussi (et peut-être avant tout) d'entrer plus avant dans la présentation de la notion d'agent, notion dont nous avons décidé de partir dans notre démarche de conception d'un système intelligent.

Le deuxième chapitre montrera le lien entre la notion d'agent et celle d'apprentissage par renforcement, et présentera l'outil mathématique choisi : les processus de décision markoviens (dont la théorie est bien maîtrisée), ainsi que leurs dérivés dans des cadres non-markoviens (lesquels seront plus utiles en pratique).

[6]Nous devrons toutefois restreindre ces présentations aux aspects qui s'avèreront utiles par la suite, faute de quoi nous risquerions de voir le lecteur se lasser plus que nécessaire, le mémoire tripler de volume, et ma thèse se prolonger d'un an.

Chapitre 1

Agentisme

Sommaire

Ce premier chapitre commencera par une section 1.1 essentielle puisqu'elle va permettre, en abordant la thématique "agent" de manière détaillée, de situer précisément le point de vue que nous adoptons dans ce mémoire. Suivra fort logiquement en section 1.2 une présentation des systèmes multi-agents qui, pour leur part, seront l'une des deux techniques sur lesquelles le travail présenté s'appuiera (l'autre étant l'apprentissage par renforcement). Le présent chapitre se terminera avec une section 1.3 consacrée à quelques points dont il nous semble important de tenir compte dans la conception d'agents ou de systèmes multi-agents.

1.1 La notion d'agent

La notion d'agent pouvant paraître quelque peu artificielle, nous allons d'abord essayer de montrer (section 1.1.1) à quel besoin elle répond, avant de revenir sur la définition d'un agent (section 1.1.2). Puis (sections 1.1.3 et 1.1.4), nous parlerons d'une part d'une classe qui nous intéressera plus particulièrement : celle des agents situés, et d'autre part de l'aspect social très souvent associé à l'agent. Pour situer plus précisément notre travail, la section 1.1.5 présentera certaines caractéristiques qui permettent de distinguer diverses approches agent. Ceci permettra de faire finalement (section 1.1.6) un rapide bilan de ce qu'est, pour nous, un agent.

1.1.1 Pourquoi cette notion ?

La partie introductive de ce mémoire a commencé par s'attacher à définir l'intelligence pour en arriver au but de l'intelligence artificielle : concevoir une entité en interaction avec un objet (on parlera plus généralement d'environnement) et essayant d'atteindre un objectif en relation avec cet objet en adaptant son comportement. Nous appellerons une telle entité un "agent" (définition qui va être précisée par la suite). Cette appellation a pour principal intérêt de faire partie d'un vocabulaire commun à nombre de travaux de recherche. Et si l'on ne reconnait pas nécessairement de but explicit ou de volonté réelle à un système, il peut toutefois être pratique de lui prêter un tel libre arbitre et de faire ainsi de "l'agentomorphisme". Comme ils l'expliquent en préface, c'est par exemple la démarche adoptée dans [Russell et Norvig, 1995], au choix de vocabulaire près qu'ils préfèrent parler d'*agent intelligent*.

On notera au passage que, en informatique, divers autres domaines font apparaître le terme "agent" [Jennings *et al.*, 1998; Chaib-draa, 1999], tels que la programmation orientée-objet, les langages d'acteur ou la conception d'interfaces homme-machine. Il ne devrait toutefois pas y avoir de confusions avec ces domaines. Encore va-t-il falloir prêter attention aux emplois variés du vocabulaire lié aux agents (comme cela est déjà apparu avec le choix propre à [Russell et Norvig, 1995] évoqué au paragraphe précédent).

1.1.2 Agent (rationnel réaliste)

Dès l'introduction de ce mémoire, nous avons choisi, parmi les quatre définitions de l'intelligence artificielle présentées par [Russell et Norvig, 1995], celle concernant les systèmes qui agissent de manière rationnelle. Nous allons suivre une partie du cheminement de ces deux auteurs, puis le resituer par rapport à quelques autres points de vue (et montrer ainsi leur diversité).

Point de vue de Russell et Norvig

Dans cet ouvrage, un agent est "tout ce qui peut être vu comme percevant son environnement à travers des capteurs et agissant sur cet environnement à travers des effecteurs", comme l'illustre la figure 1.1. Un agent humain a par exemple : pour capteurs (entre autres) ses yeux et oreilles (capteurs extéroceptifs), sans oublier les capteurs proprioceptifs (informant sur les muscles et articulations) et intéroceptifs (relatifs par exemple à la digestion) ; et pour effecteurs ses mains, jambes, bouche et d'autres parties du corps.

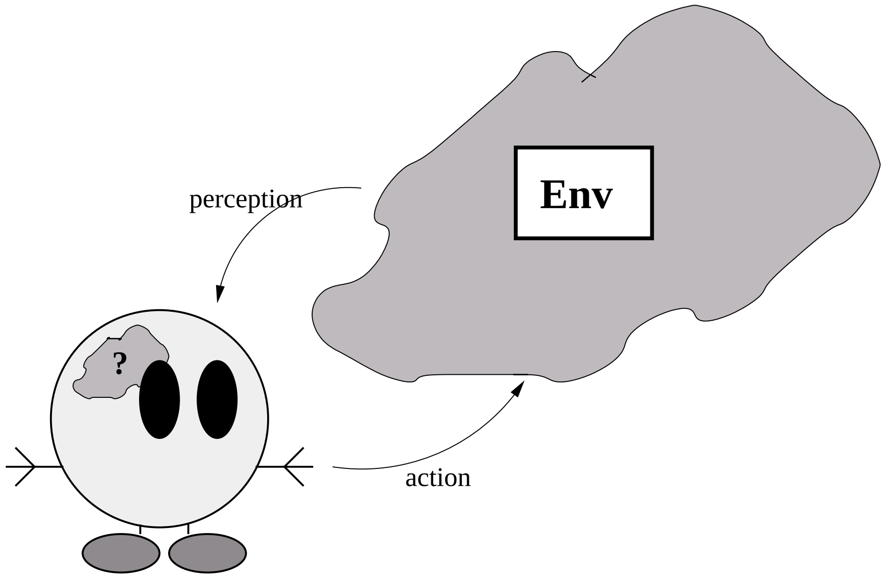


Fig. 1.1 – L'agent de Russell et Norvig interagit avec son environnement à travers ses capteurs et ses effecteurs.

Mais cette première définition ne dit rien du comportement de l'agent. Pour répondre à ce besoin, et suivant l'idée qu'un agent intelligent doit agir de manière rationnelle, un *agent rationnel* est défini comme "agissant *au mieux*", ce qui requiert une *mesure de performance* du comportement adopté par l'agent. Cette mesure va être le reflet de l'objectif de l'entité considérée. Etant dans le contexte de la conception de systèmes intelligents, cette mesure est définie par un observateur extérieur (le concepteur), mais l'agent s'en sert comme "guide" de sa propre adaptation.

Pour l'instant, on a une notion somme toute assez utopique puisque, si on la prend au pied de la lettre, elle suppose que la meilleure décision sera toujours prise. Or l'agent ne peut agir qu'étant données les connaissances qu'il a déjà acquises sur son environnement : il ne peut deviner ce qui va advenir (être doué d'omniscience). Pour cette raison va être défini un *agent rationnel réaliste*[7] comme suit :

> "Pour toute suite de percepts, un agent rationnel réaliste effectue l'action dont il est espéré qu'elle maximise la mesure de performance, et ce, sur la base de la preuve fournie par la suite de percepts et par les connaissances intégrées de l'agent."

Ainsi, on va pouvoir associer à chaque suite de percepts une action idéale, définissant de la sorte une application (appelée plus tard *politique*).

Ainsi défini, un agent rationnel réaliste répond a priori aux souhaits du concepteur d'agents intelligents. Toutefois, Russell & Norvig (désormais notés R&N) mettent aussi en avant l'*autonomie* de l'agent, c'est-à-dire le peu de connaissances a priori qu'il est utile de fournir à l'agent (modèle de l'environnement, règles comportementales...). On préfèrera un agent qui se serve le plus

[7]Les auteurs parlent d'*"agent rationnel idéal"*, mais le terme nous semble trompeur.

possible de ses expériences, plutôt qu'un agent qui requiert une spécification de comportement par le concepteur. Néanmoins, il ne semble pas raisonnable d'exiger d'un agent une autonomie complète : pour nombre de tâches possibles, ce serait pure perte de temps que de laisser l'agent acquérir de l'expérience, et cela pourrait même s'avérer dangereux. On ne peut d'ailleurs pas se passer de toute intervention extérieure, puisque la mesure de performance évoquée plus tôt est une forme de connaissance intégrée dont on ne peut se passer.

Du point de vue de Russell & Norvig, on tire donc l'idée qu'un agent est guidé par une mesure de performance, agissant dans la limite de ses moyens et expériences passées. De surcroît, on attachera un soucis particulier à l'*autonomie* de l'agent, c'est-à-dire (ici) au fait que son concepteur n'ait à fournir à l'agent qu'un minimum de connaissances relatives à sa tâche ou à son environnement.

Point de vue de Jennings et al.

R&N décrivent de manière très précise la notion d'*agent rationnel réaliste* qu'ils emploient, évitant de possibles ambiguïtés tout en se limitant au strict nécessaire. La plupart des auteurs ne s'encombrent pas de pareils soins dans leurs définitions, mais expriment des idées assez proches.

Dans [Jennings *et al.*, 1998], trois concepts-clefs sont mis en avant pour définir la notion d'agent :

- *situation* : Il s'agit ici du fait d'être en boucle fermée perception-action avec l'environnement, ce que fait un simple agent tel que défini par R&N.
- *autonomie* : L'agent décide de ses actions sans intervention d'un tiers (humain par exemple). Ce n'est pas la même idée que chez R&N, comme les auteurs le font d'ailleurs remarquer eux-mêmes : il ne s'agit plus ici de s'affranchir d'une intervention extérieure à la conception de l'agent, mais pendant son utilisation.
- *flexibilité* : L'agent répond sans délai à une nouvelle situation, et ce, de manière pro-active (de façon à atteindre un but, et non par simple réflexe). Celà correspond d'une part à l'aspect rationnel vu par R&N, et d'autre part aux contraintes de ressource (principalement temporelles). Le caractère social de l'agent est aussi mentionné, mais pourrait n'être vu que comme un résultat de la présence d'autres agents.

En fin de compte, on retrouve dans ce deuxième point de vue des idées très proches de celles de R&N et, globalement, les mêmes aspects principaux sont présents. Il apparaît toutefois génant que des désaccords existent, tels que celui concernant le terme “autonomie”.

Notre définition

Nous rencontrerons plus tard une troisième définition. Mais de ce qui vient d'être présenté, nous tirons pour l'instant la définition du terme “agent” que nous suivrons désormais : nous appellerons *agent* ce que Russell & Norvig appellent agent rationnel réaliste, en précisant le fait que les décisions sont prises sous contraintes de ressources (temps machine, espace mémoire...), et en insistant sur l'importance que revêt le caractère autonome d'un agent.

1.1.3 Agent situé

Si l'on a exprimé ce qu'est un agent, on a peu dit pour l'instant de la manière de le concevoir, puisqu'il n'y a pas de précisions données a priori sur ce point. Un système expert à base de règles comme un système fondé sur des algorithmes génétiques ou toute autre approche peuvent servir à la réalisation d'un agent. Toutefois, nous allons contraindre la forme des agents, ou plus précisément nous restreindre à l'utilisation de certaines techniques de l'intelligence artificielle, et ce, pour deux raisons étroitement liées l'une à l'autre : la gestion de l'incertitude et le problème de l'ancrage des symboles (ceci nous amènera en fait à la notion d'agent situé).

Incertain

Une première raison pour se restreindre à certaines techniques de l'IA est l'incertitude dans laquelle peut vivre un agent. Comme nous l'avons déjà évoqué, elle peut intervenir à différents moments :

- Les perceptions peuvent être partielles, floues, et donc prêter à diverses interprétations.
- Le résultat des actions peut être incertain, du fait d'une précision limitée des effecteurs.
- L'évolution de l'environnement peut être inattendu si la connaissance d'un modèle du monde est imparfaite.

Tout ceci amènerait au moins à exclure les systèmes de l'intelligence artificielle symbolique classique utilisant une logique simple dans laquelle la prise en compte de l'incertain s'avère complexe. Au contraire, la logique floue [Zadeh, 1973] ou la logique possibiliste [Dubois *et al.*, 1994] (par exemple) peuvent tenir compte de connaissances imprécises, même si leur expressivité reste limitée.

L'intérêt du passage par des symboles est de permettre des manipulations très efficaces, des raisonnements assez élaborés aboutissant à des systèmes de planification ou de diagnostic dans des domaines complexes. Mais cela ne suffira pas à nous faire adopter de telles approches.

Ancrage des symboles

Une deuxième raison pour contraindre les techniques d'IA que nous comptons utiliser est liée à l'autonomie de l'agent, autonomie que nous cherchons à favoriser. Or, une connaissance bien souvent apportée par la main du concepteur est celle de symboles (nous y revenons), comme illustré par la figure 1.2.

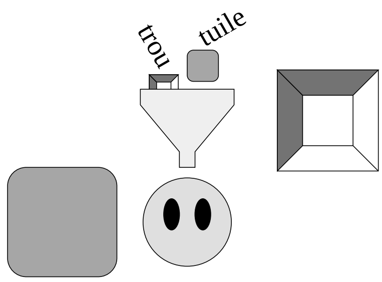


FIG. 1.2 – Agent “nourri” avec des symboles utiles au monde des tuiles.

La conception d'un agent par intelligence artificielle symbolique se base sur l'hypothèse qu'un ensemble adéquat de symboles peut être déterminé et utilisé pour raisonner sur l'environnement réel. Cette première hypothèse est déjà forte, et représente une intervention importante du concepteur. Un deuxième aspect notable est l'ancrage de ces symboles par rapport aux perceptions et actions de l'agent : un tel système de symboles utilisé dans un agent doit avoir une interprétation sémantique, être relié au monde réel, ce qui n'est pas sans difficultés (sur le sujet, voir [Harnad, 1990]). A propos de ce problème, [Brooks et Stein, 1994][8] explique que la

[8][Jennings *et al.*, 1998] fait remarquer que, dans les papiers de Rodney Brooks abordant ce thème, l'approche de l'I.A. qu'il propose est appelée de manière variée *I.A. comportementale*, *I.A. réactive* et *I.A. située* [Brooks, 1986; 1991b; 1991a].

"bonne vieille I.A." a longtemps négligé le problème réel que pose l'implémentation des symboles : comment les construire.

Ainsi, même si la gestion de l'incertitude n'était pas un problème dans des approches symboliques, l'utilisation de symboles elle-même suppose un apport de connaissances trop important pour être raisonnable par rapport à notre souhait d'autonomie (au sens de R&N).

Conclusion

L'incertitude que doit gérer l'agent et l'intervention importante que suppose la définition de symboles nous feront exclure les systèmes basés sur l'I.A. symbolique. Nous travaillerons sur ce que l'on convient d'appeler des *agents situés*[9].

Si l'on reprend les idées de [Vera et Simon, 1993] ou [Clancey, 1997], un agent est situé s'il utilise des représentations externes plutôt qu'internes, rien ne valant l'expérimentation réelle à travers la boucle perception-action. Néanmoins il n'y a pas, sous cet angle, d'interdiction stricte de la notion de symbole ou de toute autre représentation interne, mais rien n'est fixé à l'avance.

Nous allons définir un agent situé comme n'ayant pas a priori de système de représentation interne spécifié par le concepteur (puisqu'on souhaiterait se passer de l'intervention de ce dernier). Par contre, on n'exclut pas que de telles représentations apparaissent d'elles-même au sein de l'agent (imaginez un agent qui découvre la notion de pomme, de nombre...).

1.1.4 Agent social

A travers la définition du concept de flexibilité (dans la définition du terme "agent" par [Jennings *et al.*, 1998]) est déjà apparue l'idée qu'un agent peut être en présence d'autres entités intelligentes, et doit être en conséquence "social". Ce qualificatif désigne le fait que l'agent "doit interagir de manière appropriée avec d'autres agents artificiels ou humains[10] de façon à accomplir sa propre résolution de problème et d'aider les autres dans leurs activités."

Cet aspect social peut être vu comme un résultat attendu chez un agent tel que nous l'avons défini, lequel va pouvoir apprendre à coopérer avec autrui (voir figure 1.3). Pour cette raison, nous n'avons pas, pour notre part, retenu cette caractéristique comme essentielle chez l'agent. Toutefois, notons que c'est un aspect qui s'avère souvent intéressant, que cela concerne des agents en simple voisinage et sans relations importantes ou des sociétés au sein desquelles apparaissent des phénomènes de fortes collaborations.

Une autre définition du terme "agent" dans laquelle le caractère social est particulièrement mis en avant est celle de [Ferber, 1995], puisqu'y sont cités directement la communication entre agents (de manière directe ou indirecte), l'offre de services à d'autres agents, et l'éventuelle capacité de reproduction. Mais cela se justifie principalement par l'importance du cadre multi-agents dans ses travaux.

Nous en resterons là pour l'instant sur la vie de groupe des agents, en retenant principalement que le caractère social est, non pas une caractéristique fondamentale de l'agent, mais plutôt un effet naturellement induit par la quête de son objectif : l'interaction avec d'autres n'est qu'un moyen possible. La deuxième partie de ce premier chapitre (section 1.2) y reviendra en détail en développant le domaine des systèmes multi-agents.

[9]Attention : Encore une fois cette expression rencontre des réalités différentes selon les auteurs. On s'éloigne d'ailleurs ici sensiblement de ce qui est appelé "situation" dans [Jennings *et al.*, 1998] et que l'on a rencontré précédemment.

[10]Disons plutôt "agents naturels".

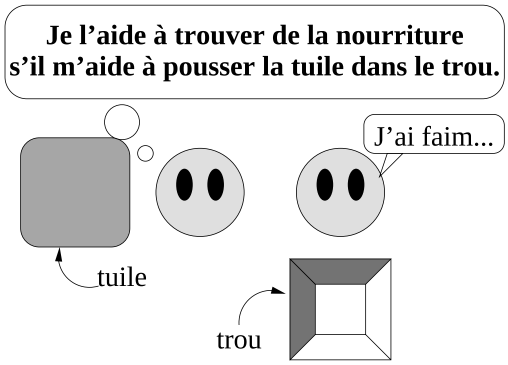


FIG. 1.3 – Agent (à gauche) raisonnant sur le moyen d'atteindre son but en l'échange d'un service rendu.

1.1.5 Autres critères de différenciation entre agents

Nous avons résolu, pour le cadre de ce mémoire, le problème de définir la notion d'agent, et avons aussi mis en avant deux aspects (situé et social) qui resteront prépondérants dans notre travail. Nous allons dans la présente section nous attarder sur d'autres caractéristiques classiques des agents, lesquelles vont permettre de préciser sous quelles hypothèses est conduite cette thèse.

1.1.5.1 Environnement

L'interaction agent/environnement étant essentielle dans le problème que pose l'intelligence artificielle, nous commencerons ici par présenter une caractérisation relativement générique de l'environnement, du point de vue de l'agent, laquelle est issue des travaux de [Russell et Norvig, 1995]. Les cinq principales propriétés qu'il est proposé d'observer sont :

- **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 ? Remarque : un environnement déterministe peut ne pas l'être *du point de vue de l'agent*, du fait d'une connaissance insuffisante de l'état de l'environnement.
- **Episodique / non-épisodique** : On parle d'environnement épisodique (on dit aussi que le temps est discrétisé) si l'expérience de l'agent peuvent être divisée en “épisodes”. Chaque épisode consiste en une phase de perception puis en une phase d'action, le résultat de l'action ne dépendant que de la période courante.
- **Statique / dynamique** : Un environnement est dit dynamique s'il peut changer pendant la prise de décision de l'agent[11]. Il est dit statique dans le cas contraire. Si le seul changement lié au temps est la performance atteinte, on parlera d'environnement semi-dynamique.
- **Discret / continu** : S'il existe un nombre limité de perceptions et d'actions possibles, on parlera d'environnement discret.

On notera que ces distinctions correspondent souvent à des distinctions entre situation réelle et situation idéale. S'il est pratique de faire l'hypothèse que l'environnement est épisodique et discret, ce n'est pas le cas du monde perçu par un chauffeur de taxi.

[11] Attention : dans la suite de l'exposé, un environnement sera dit dynamique si les règles de son fonctionnement évoluent.

Du fait des techniques employées dans cette thèse (plus particulièrement les techniques d'apprentissage par renforcement), nos agents perçoivent leur environnement en général comme : inaccessible (perceptions partielles), non-déterministe (du fait d'abord de l'inaccessibilité), épisodique, statique, et discret.

La section suivante revient sur un cas particulier lié à la notion d'accessibilité.

1.1.5.2 Localité

Ce qui a été vu jusqu'à présent sur le concept d'agent ne suppose pas l'existence d'une notion de distance, d'espace doté d'une quelconque topologie. Pourtant, on se place assez souvent dans un tel cadre de manière implicite, certaines définition du terme "agent" supposant même l'existence d'une topologie. Ceci est, entre autres, lié au fait que nombre de problèmes placent des agents dans un espace vectoriel à deux ou trois dimensions, ou dans un graphe (tel qu'un réseau informatique, le web...).

De ce fait, il n'est pas rare qu'apparaisse la notion de localité (des actions et des perceptions) sans que soit mentionné le fait qu'elle n'a de sens qu'à condition que l'idée de voisinage en ait un. Pour notre part, nous distinguerons deux formes de localité :

- *localité (classique)* : On pourrait aussi parler de localité au sens fort. Ici, on suppose que les perceptions et actions ne sont possibles que dans un rayon strictement défini autour de l'agent (voir figure 1.4). Il est à noter que, à l'intérieur de cette zone, les perceptions peuvent n'être que partielles et les actions imprécises.

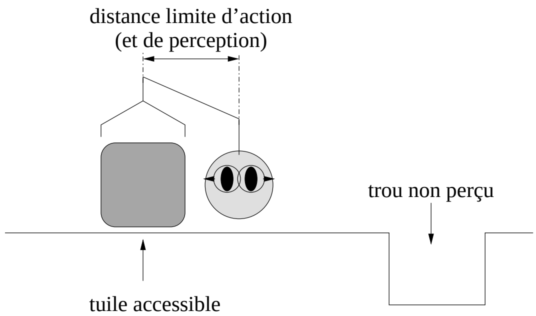
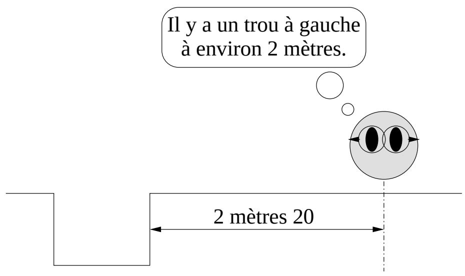


FIG. 1.4 – *localité* : les perceptions et actions de l'agent sont limitées à un certain rayon d'action.

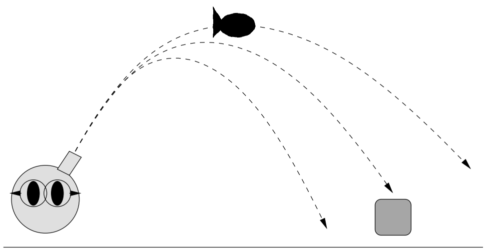
- *subjectivité [Chadès, 2003]* : Par opposition, c'est une localité au sens faible. Si elles sont possibles à toutes distances (ce qui n'est pas une obligation), perceptions et actions voient ici leur qualité se dégrader avec l'éloignement (voir figure 1.5).

Remarque : La notion de topologie est particulièrement présente dans les cadres multi-agents, au point qu'il peut être difficile de s'imaginer un tel système dénué de topologie. Un exemple qui semble toutefois possible est celui des architectures à tableau noir [Hayes-Roth, 1985], dans lesquelles des entités résolvent ensemble un problème en écrivant leurs résultats partiels sur un tableau commun. On pourrait même imaginer un tel tableau sur lequel les agents aient plus ou moins de mal à lire les notes de leurs congénères (pour montrer qu'on peut avoir des perceptions partielles).

Pour information, si nos travaux ne s'attachent pas particulièrement à la localité des agents (cette propriété n'intervenant pas directement dans les méthodes proposées), la plupart des exemples que nous traiterons se placeront dans le cadre d'agents subjectifs tels que décrits précédemment.



a- myopie : distance perçue approximativement.



b- imprécision d'une action de tir.

FIG. 1.5 – *subjectivité* : les perceptions (a) et actions (b) de l'agent sont meilleures à courte distance.

1.1.5.3 Architecture interne

Outre la relation agent/environnement, un deuxième aspect important du problème de conception d'un agent est de décrire son architecture. On parle ici de l'architecture interne d'un agent (le mécanisme qui régit son comportement) par opposition à l'architecture externe (les capteurs et effecteurs dont il est doté) dont il ne sera question qu'en section 1.3.1.1.

L'architecture interne d'un agent va être un point essentiel dans sa conception. Un agent a été présenté comme devant adopter un comportement adaptatif et être autonome (au sens de R&N). Le travail du concepteur est donc de créer un système qui *gère* un tel comportement, et dont le comportement n'est que le résultat. Ce "système" est ce que nous appelons architecture interne, et constitue a priori le cœur d'un agent tel que nous l'avons défini. Une partie de nos travaux propose une telle architecture, ce qui justifie la rapide présentation qui suit (et à laquelle il sera fait référence ultérieurement).

Réactif/cognitif

Dans les distinctions faites au niveau des architectures internes, l'opposition principale est faite entre les agents dits réactifs, pouvant fonctionner par exemple selon le schéma stimuli-réponses, et les agents dits cognitifs, tels que ceux capables de planifier sur la base d'un modèle de leur univers.

La limite entre ces deux notions est imprécise. Un agent conçu sur la base d'un réseau de neurones peut être vu comme un simple automate, mais si un apprentissage de type essais-erreurs lui permet de s'adapter de manière efficace, un système de raisonnement utilisant une représentation interne de l'environnement peut s'élaborer.

Ayant choisi d'orienter nos travaux vers des agents situés, on peut considérer que nos agents sont réactifs. Mais, même si elle est courante (au point de ne pouvoir raisonnablement être passée sous silence), cette caractérisation nous servira peu.

Quelques architectures types

On peut trouver dans la littérature nombre d'architectures proposées pour la conception d'agents : le survol du sujet effectué dans [Müller, 1997] le montre bien. Nous allons simplement citer quatre formes d'architectures modulaires qui serviront ultérieurement (en section 8.3.2) de point de comparaison avec nos travaux :

1. **Architecture horizontale** [Ferguson, 1992] : Ici, on suppose que n modules indépendants existent, chacun motivé par un but propre et capable de fonctionner de manière

autonome, c'est-à-dire de gérer, s'il était seul, du traîtement des perceptions au choix des actions. L'intérêt principal de cette approche est de pouvoir utiliser des modules conçus indépendamment les uns des autres.

Mais un problème essentiel réside dans la mise au point d'un contrôleur décidant de l'action à réaliser effectivement parmi celles proposées par les différents modules : il faut définir une manière de combiner ces modules en gérant les priorités entre ceux-ci. La figure 1.6 présente une telle architecture, sans pour autant qu'apparaisse le contrôleur qui vient d'être cité.

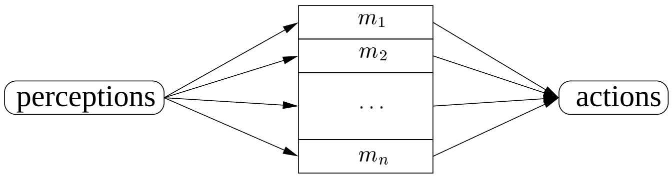
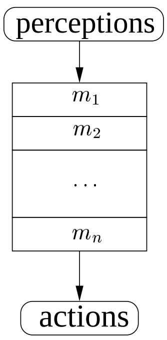


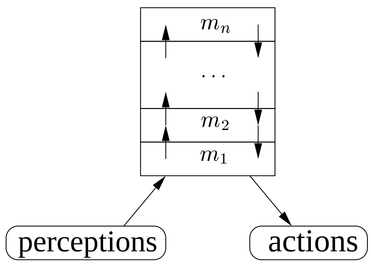
FIG. 1.6 – Principe d'une architecture horizontale.

2. **Architecture verticale** :
   (a) **à une passe** : Le traîtement de l'information passe dans des modules successifs de l'entrée (côté capteurs) à la sortie (côté effecteurs). C'est le fonctionnement d'un robot planificateur classique avec par exemple m_1 =perception, m_2 =modélisation, m_3 =planification, m_4 =exécution de tâche, et m_5 =contrôle moteur.
   (b) **à deux passes** [Müller et Pischel, 1993] : Dans ce cas, on travaille par niveaux d'abstraction successifs. Un seul module (noté m_1) est en lien direct (perceptions et actions) avec l'environnement réel. Ce module m_1 produit des perceptions abstraites pour un second module m_2 dont les actions vont être des commandes pour le m_1. On peut ainsi empiler un nombre quelconque de modules.

   La figure 1.7 montre les schémas de principe de ces deux architectures. Elles ont pour principal défaut de supposer l'existence de plusieurs niveaux d'abstraction, donc l'utilisation de symboles prédéfinis. Ceci va à l'encontre du caractère "situé" que nous cherchons à respecter.



a- architecture verticale à une passe.



b- architecture verticale à deux passes.

FIG. 1.7 – Principe d'une architecture verticale (a) à une passe et (b) à deux passes.

3. **Architecture à subsomption** [Brooks, 1986; 1989] : Il s'agit dans cette architecture de mettre en interaction des modules indépendants, ayant chacun un rôle propre. Chaque module agit selon les perceptions qu'il reçoit en entrée, ses commandes en sortie permettant :

soit de contrôler directement des effecteurs, soit d'inhiber des sorties d'autres modules, soit de supprimer des entrées. Deux aspects qui me semblent importants peuvent être ici notés :

– *Par rapport aux architectures précédentes* : On peut voir cette architecture à subsomption comme une généralisation des précédentes : dans le principe, elle peut reproduire le même fonctionnement que chacune des trois autres. En étant simplificateur, on est avec cette dernière architecture dans un système de réseau de modules dont les trois précédentes ne sont que des cas particuliers (au problème près de l'architecture horizontale et de son contrôleur).
– *Conception incrémentale* : Une des idées soutenues dans la présentation de cette architecture par Brooks est que l'on peut commencer par concevoir un système assez simple et l'augmenter de manière incrémentale pour répondre à des besoins de plus en plus complexes, les nouveaux modules "subsumant" les précédents. Il est à noter que c'est aussi vrai pour l'architecture verticale à deux passes (pas pour l'architecture horizontale, puisque le contrôleur pose problème[12]).

Cette dernière architecture, très générique, a évolué à travers divers travaux. Notre présentation en restera aux premières versions de Brooks. Elles semblent offrir de nombreuses possibilités, mais n'offrent pas de moyen de conception automatisée.

Cette rapide présentation montre que les approches pour définir l'architecture d'un agent sont diverses, mais que ce problème central est loin d'être résolu : si la proposition de Brooks semble permettre des constructions assez génériques, l'automatisation de cette conception reste une question ouverte.

Nous ne dirons pas ici quelle forme d'architecture nos agents adopterons. C'est justement un aspect qui pourra prêter à discussion.

Digression

Pour conclure sur ce point, on peut remarquer que ces quatre approches ne s'excluent pas toutes mutuellement puisque, par exemple, un module d'une architecture horizontale pourrait très bien être lui-même formé d'une architecture verticale complète (et vice-versa). Ce genre de manipulation pourrait tout autant faire intervenir une architecture à subsomption. Comme on l'a dit, il s'agit de réseaux de modules (à rapprocher de systèmes connexionistes ?), chaque module pouvant lui-même être un tel réseau de modules. Certes, un réseau de réseaux est un réseau, mais la structuration en deux niveaux différents (deux échelles) peut avoir l'intérêt, pour un œil extérieur tel que celui du concepteur, de rendre l'ensemble plus compréhensible.

Cette dernière digression peut sembler un peu incongrue, mais les travaux présentés par la suite feront apparaître des remarques y faisant écho.

1.1.5.4 Mémoire et communication

Cette présentation du concept d'agent serait incomplète sans parler de mémoire et de communication, deux données importantes du problème de conception d'un agent. Comme nous allons maintenant le voir, ces deux termes prêtent parfois à discussion, du fait de possibles ambiguïtés.

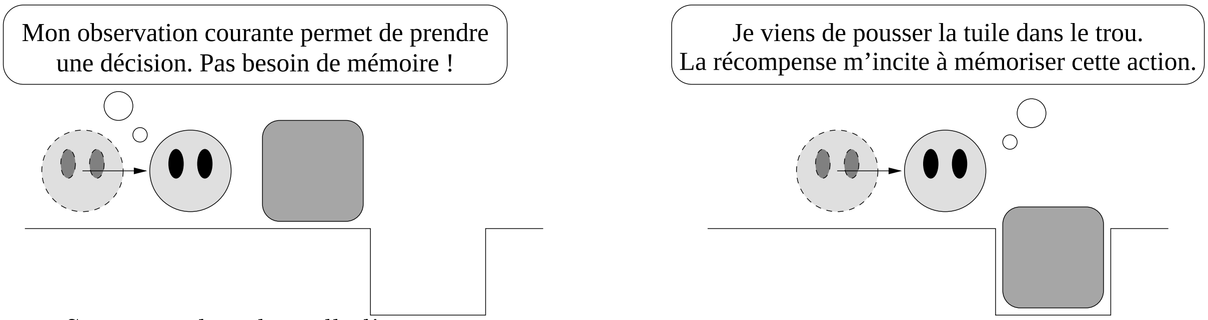
Avec/sans mémoire

Un premier point est que, pour simplifier la tâche des agents que nous allons concevoir, nous ferons l'hypothèse qu'ils peuvent décider des actions à effectuer uniquement d'après l'observation

[12] Oui, je radote.

immédiate de leur environnement. Si les perceptions accessibles et l'environnement s'y prêtent, cela sera suffisant pour avoir un comportement optimal. Mais, en général, l'efficacité de l'agent en sera réduite.

On parlera alors d'agents sans mémoire en sous-entendant "sans mémoire à court terme". En effet, comme les méthodes de conception utilisées se fondent sur une adaptation par apprentissage, il s'agit d'une forme de mémorisation à long terme (mémorisation des effets du comportement de l'agent en l'occurence). Ces deux échelles de mémoire sont présentées sur la figure 1.8.



a- Situation dans laquelle l'agent peut choisir son action (pousser la tuile dans le trou) sans se souvenir de ce qu'il vient de faire.

b- L'agent vient, par hasard, de pousser la tuile dans le trou. Il voit que cela est bon, et décide de s'en souvenir.

FIG. 1.8 – Illustrations d'un agent (a) sans mémoire à court terme, mais (b) avec mémoire à long terme.

Pour notre part, on choisira de travailler sans mémoire à court terme. Utiliser plus que les perceptions immédiates serait très coûteux.

Communiquant ou non

Le second point concerne l'aspect social de l'agent. Dans le cas où il est en présence de congénères, nous ferons le choix de ne pas faire communiquer *explicitement* les agents entre eux. Il n'existera pas d'actions et de perceptions dédiées à une forme de communication, mais les actions d'un agent A pourront toujours être perçues par un agent B, lequel pourra agir en conséquence, ainsi que l'illustre la figure 1.9.

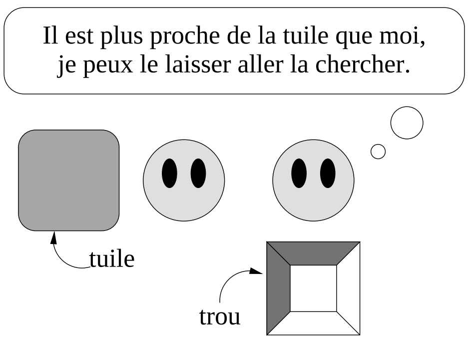


FIG. 1.9 – On peut voir la présente situation comme impliquant une communication indirecte : parce qu'il est près de la tuile, l'agent A (à gauche) signifie à l'agent B (à droite) son intention probable d'aller la chercher.

Le fait d'avoir des actions et perceptions dédiées à la communication entre agents amènerait sûrement de riches possibilités pour améliorer leur coopération, mais au prix d'une complexification importante de la conception. Nous laisserons donc ces possibilités de côté dans ce mémoire.

1.1.6 Bilan

Dans cette section, nous avons d'abord essayé de préciser la définition du terme "agent", en faisant référence à diverses sources, dont principalement R&N, pour orienter notre position vers celle des agents situés. A propos du caractère social des agents, nous avons exprimé le fait que c'est pour nous plus un effet de bord qu'une caractéristique intrinsèque. Par contre, nous avons décrit un certain nombre d'autres caractéristiques qui, si elles ne seront pas particulièrement étudiées dans la suite de ce manuscrit, doivent être précisées afin de bien situer le cadre de travail choisi. On peut ainsi tirer le rapide bilan suivant sur ce que seront les agents rencontrés dans la suite de ce manuscrit :

> Désormais, on considérera des agents :
> – rationnels idéaux [Russell et Norvig, 1995] et aussi autonomes que possible,
> – situés (sans représentation interne a priori),
> – éventuellement considérés pour leur caractère social,
> – sans mémoire à court terme (mais à long terme), et
> – sans moyens de communication directe.
>
> C'est plus particulièrement à l'architecture interne que nous nous intéresserons, en tant que cœur de l'agent.
>
> De plus, un agent verra ici son environnement comme :
> – inaccessible[a],
> – non-déterministe,
> – épisodique,
> – statique, et
> – discret.
>
> [a] On parlera de perceptions partielles, ce qui, dans nos applications, se traduira par des agents subjectifs.

Ceci fait, nous allons pouvoir nous attarder maintenant sur un domaine qui prend appui sur l'utilisation d'agents en groupe, celui des systèmes multi-agents.

1.2 Systèmes Multi-Agents

Dans la section précédente, nous avons déjà fait remarquer que nombre d'auteurs, quand ils donnent une définition du mot "agent", ont déjà en tête le fait que d'autres agents seront là avec lesquels interagir. Pour notre part, nous considérons simplement que des agents peuvent développer un caractère "social" s'il aide à atteindre leurs objectifs.

Mais la mise en présence de plusieurs agents va aussi nous intéresser pour obtenir une *résolution "collective" de problèmes*, que ce soit une nécessité (quand le problème posé contraint l'utilisation d'un tel groupe), ou que ce soit un moyen (quand on préfère attaquer un problème à l'aide d'un ensemble d'entités intelligentes).

Nous allons commencer par voir plus précisément quel est ce domaine des systèmes multi-agents (SMA) en essayant d'abord de le définir en section 1.2.1, puis à travers la présentation d'un exemple au cours de la section 1.2.2. S'il y a en fin de compte peu de théorie permettant de bien maîtriser les phénomènes étudiés dans ce domaine, on verra en section 1.2.3 un certain nombre de concepts des plus utiles à l'analyse d'un SMA, concepts qui aideront aussi à comprendre les mécanismes intervenants dans un SMA et les questions soulevées dans ce domaine.

1.2.1 Définitions rencontrées

RDP+SMA=IAD

[Jennings *et al.*, 1998] (dont nous nous inspirons ici fortement) expliquent que la recherche sur les systèmes composés de plusieurs agents était classiquement dénommée *intelligence artificielle distribuée (IAD)* et divisée en deux camps [Bond et Gasser, 1988] :

- la **résolution distribuée de problèmes** (RDP) : Elle cherche à résoudre un problème donné à l'aide de modules (nœuds) qui coopèrent en divisant et partageant la connaissance sur le problème et ses solutions en cours d'élaboration. Les stratégies d'interaction font partie intégrante du système considéré.
- les **systèmes multi-agents** (SMA) : Par opposition, la recherche sur les SMA s'intéresse au comportement d'un ensemble d'agents (autonomes, au sens de [Jennings *et al.*, 1998], puisqu'il n'y a pas de système de contrôle global) dont le but est aussi de résoudre un problème donné. On peut voir un SMA comme un réseau "faiblement couplé" de résolveurs de problèmes qui s'attaquent à des difficultés au-delà de leurs capacités ou connaissances individuelles.

Pour résumer de manière peut-être abusive, la RDP distribue une tâche au sein d'un ensemble organisé de "modules", là où les SMA se fondent sur les plus grandes capacités d'entités en interaction libre.

Il est à noter que, récemment, le sens du terme "système multi-agents" s'est étendu pour être utilisé pour tout type de systèmes constitués de composants (semi-)autonomes [Jennings *et al.*, 1998]. Il semble que, d'une part, la recherche propre à la RDP est peu développée, puisque tout système modulaire et centralisé en fait partie, et que, d'autre part, l'assouplissement de la notion de SMA lui fait aujourd'hui recouvrir la part active majeure de l'IAD. Ici, nous présentons les SMA dans leur forme classique, et préciserons quand notre propos s'en éloigne.

Pour resituer ce qu'est un SMA par rapport à ce qui a déjà été dit de l'agent, précisons que nombre des travaux dans le domaine des SMA s'intéresse à l'aspect résolution collective de problème en mettant en jeu des automates et non nécessairement des agents au sens que nous avons choisi. La présente introduction aux SMA s'efforcera de distinguer, quand cela s'avérera pertinent, les deux cas de figures présentés.

Modèle influences-réaction

Le modèle influences-réaction proposé par [Ferber et Müller, 1996] me semble intéressant pour comprendre le fonctionnement d'un SMA. On se base sur l'idée que les agents exercent une *influence* sur leur environnement, lequel répond par une *réaction*, d'où le nom de modèle *influences-réaction*. C'est toutefois une version modifiée de ce formalisme, inspirée de celle proposée par [Bouzid, 2001], que nous présentons. Le modèle en question définit un SMA comme étant un tuple $\langle Ags, E, \Pi \rangle$, où :

- Ags est l'ensemble des agents.
- E est l'environnement.
- Π est l'opérateur de combinaison des influences simultanées des agents.

Deux types de sous-systèmes dynamiques apparaissent ici, dont voici le détail :

1. **L'environnement** E est défini par le tuple $\langle A, S, T \rangle$ dans lequel :

- $A = \bigcup_{a \in Ags} A_a$ est l'ensemble des influences que peuvent entreprendre les agents.
- S est l'ensemble des états possibles de l'environnement ($s(t)$ état de E à l'instant t).
- T définit les lois d'évolution de l'environnement :

$$s(t+1) = T\left(s(t), \Pi_{a \in Agents} Infl_a(\xi_a(t))\right).$$

2. **Chaque agent** a est défini par un tuple $\langle Percept_a, P_a, \Xi_a, F_a, A_a, Infl_a \rangle$ dans lequel :
   - $Percept_a$ est l'ensemble des percepts possibles de l'agent.
   - P_a est la fonction de perception de l'agent.
   - Ξ_a est l'espace des états internes de l'agent ($\xi_a(t)$ état interne à l'instant t).
   - F_a définit la dynamique interne de l'agent par (avec $s(t) \in S$) :

$$\xi_a(t+1) = F_a\left(\xi_a(t), P_a(s(t))\right).$$

   - A_a est l'ensemble des influences que peut exercer l'agent.
   - $Infl_a$ est la fonction de décision de l'agent. Elle associe à un état interne $\xi_a(t)$ une influence de A_a.

L'**opérateur** Π, pour sa part, permet de combiner les influences des différents agents et d'en déduire l'effet réel du groupe sur l'environnement. Cet opérateur est particulièrement important, dans la mesure où c'est grâce à son rôle qu'un groupe d'agents n'est pas qu'une accumulation d'entités. C'est dans Π que s'opèrent les interactions entre agents, qu'apparaissent des problèmes de concurrence ou de coopération. Pour résumer, c'est grâce à lui qu'un système multi-agents est plus que la somme de ses parties (ce qui correspond à la définition philosophique du holisme).

Comme le montre la figure 1.10, ce modèle permet de voir un système multi-agents comme un ensemble de boucles comparables à la boucle perception-action vue en section 1.1.2 (figure 1.1) pour présenter un agent seul en interaction avec son environnement. La principale différence à laquelle il faut faire attention est qu'on ne considère pas ici directement les actions des agents comme telles, puisque les actions de deux agents peuvent être conflictuelles. D'où le principe d'exprimer les comportements des agents sous la forme d'*influences*, dont la résultante (calculée par Π) va, elle, agir sur l'environnement.

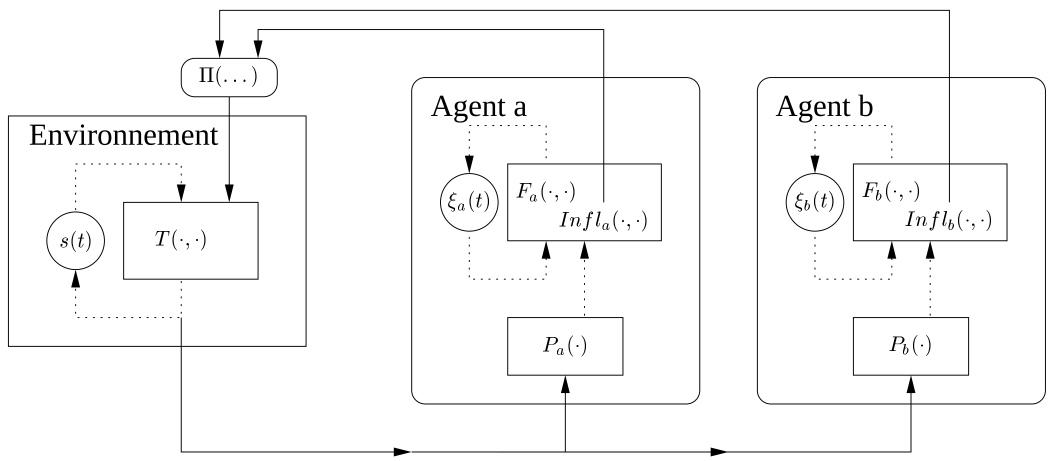


FIG. 1.10 – Modèle influences-réaction à deux agents.
Les points entre parenthèses indiquent que les paramètres sont les entrées de chaque "boîte".

On peut finalement faire quelques remarques :

- Tout d'abord, on considère ici un système à *temps discret* dans lequel tout le système fonctionne de manière simultanée, alors que l'aspect asynchrone des SMA est souligné par certains, tels que [Jennings *et al.*, 1998]. C'est pour des raisons pratiques que ce choix est fait. D'ailleurs pour notre part nous manipulerons principalement des SMA vérifiant cette propriété de fonctionnement à temps discret.
- D'autre part, le problème des *objectifs* des agents n'est pas évoqué. Si l'on adopte le point de vue que chaque agent a connaît son objectif propre (éventuellement sous forme de récompenses), c'est derrière les fonctions $Infl_a$ et F_a que se cache l'expression de ce but.
- Pour ce qui est des *communications* entre agents, elles passent nécessairement par l'environnement, même s'il n'y a pas nécessairement d'effet persistant sur celui-ci (modifications à long terme telles qu'une coloration).
- Enfin, l'*opérateur* Π reste plutôt mystérieux. Il ne paraît pas évident de tracer la limite entre son rôle de gestion des interactions entre influences et le rôle de la fonction de transition T.

Cela reste toutefois un modèle qui illustre bien le fonctionnement d'un système multi-agents, en le présentant comme un couplage de systèmes dynamiques.

1.2.2 Un exemple

Pour montrer les propriétés que l'on cherche à faire ressortir d'un SMA, et plus particulièrement le fait qu'un groupe d'agents est plus que la simple somme de ses éléments, nous allons maintenant donner un exemple de problème d'optimisation résolu par une approche SMA d'inspiration biologique : la recherche de plus court chemin par une colonie de fourmis. Il n'est pas question ici d'agents rationnels mais plutôt d'automates, cet exemple ayant pour but d'illustrer l'idée d'un comportement de groupe complexe issu d'interactions simples.

Ce sont des travaux d'éthologues sur les comportements collectifs d'insectes sociaux (fourmis, abeilles, araignées, termites...) qui ont abouti à la mise au point d'algorithmes tels que celui-là, lequel a été présenté pour la première fois comme une approche de calcul distribué dans [Dorigo *et al.*, 1991]. De là sont apparus un certain nombre d'applications réelles des algorithmes fondés sur une telle forme d'intelligence en essaim que sont ACO (Ant Colony Optimization) et ACR (Ant Colony Routing) [Bonabeau *et al.*, 1999].

Nous allons brièvement montrer le principe de l'expérience menée, le fonctionnement du phénomène d'optimisation lui-même (cette partie purement éthologique étant due à [Deneubourg *et al.*, 1990]) et enfin parler de la réutilisation de cet algorithme.

Expérience

L'expérience biologique en elle-même est relativement simple : elle consiste en une colonie de fourmis, en une source de nourriture, et en deux chemins A et B de longueurs différentes mais menant tous deux de la colonie à la nourriture (on supposera que A est le plus court). La figure 1.11 résume ces données de manière schématique.

On observe au cours du temps la proportion de fourmis empruntant l'une et l'autre branche. Au début, les deux parcours possibles pour atteindre la nourriture et retourner au nid sont utilisés de manière égale. Puis, pendant une phase transitoire, les fourmis se mettent progressivement à emprunter de plus en plus le chemin le plus court, pour finir par s'en contenter et presque abandonner l'autre option.

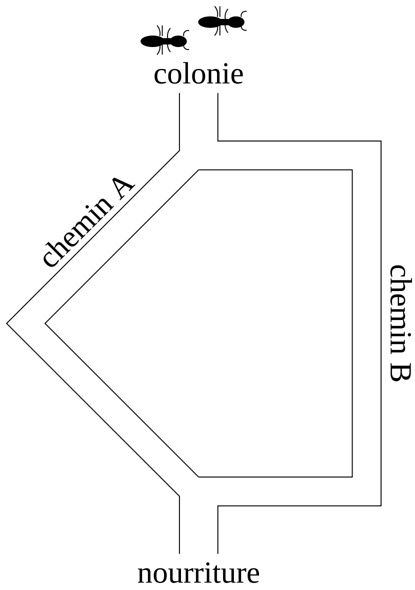


FIG. 1.11 – Dispositif de l'expérience menée avec la colonie de fourmis

Explication

Ce sont deux caractéristiques des fourmis sociales considérées qui vont permettre d'expliquer ce phénomène : d'une part, quand elles parcourent un chemin entre nourriture et nid, les fourmis déposent des traces chimiques (phéromones) dans leur environnement, lesquelles tracent s'évaporent lentement ; et d'autre part, une fourmi a toujours tendance à suivre les fortes concentrations en phéromone.

Dans l'expérience qui nous intéresse, l'environnement étant vierge de toute trace chimique au départ, les insectes n'ont pas de préférence pour l'une ou l'autre voie. Par contre, le dépot de phéromones est plus vite effectué sur la voie la plus courte, puisqu'elle permet un parcours plus rapide (phénomène illustré par la figure 1.12). Ainsi commence un effet boule de neige : le déséquilibre de concentration se renforce du fait de l'attirance croissante des fourmis.

Il est intéressant de noter que la notion de distance n'apparaît nul part explicitement dans ce phénomène. Un autre aspect est l'utilisation de l'environnement comme support pour "communiquer" entre consœurs (on parlera de *stigmergie* [Grassé, 1959][13]) comme pour dessiner la solution de leur problème à force de retravailler l'ébauche de départ (il y a *émergence* du plus court chemin). En somme, on a bien ici un phénomène lié au groupe même, et non à la simple présence de multiples agents.

Réutilisation

L'article [Dorigo et Di Caro, 1999] présente l'algorithme Ant Colony Optimization qui est à l'origine de ces applications. Il ne se base pas sur une stricte simulation réaliste du comportement des fourmis, lequel ne se prêterait pas à la réutilisation. En effet, il est par exemple difficile de reproduire le phénomène des déplacements asynchrones (d'autant qu'il serait une pure perte de temps que de simuler les temps de parcours). En revanche, une fourmi artificielle va pouvoir, après parcours d'un chemin, déposer a posteriori des "phéromones" en quantités inversement proportionnelles au coût de ce chemin. On remplace donc la différence de temps mis pour parcourir

[13]La stigmergie peut être décrite comme la "stimulation des ouvriers par la performance qu'ils ont accompli".

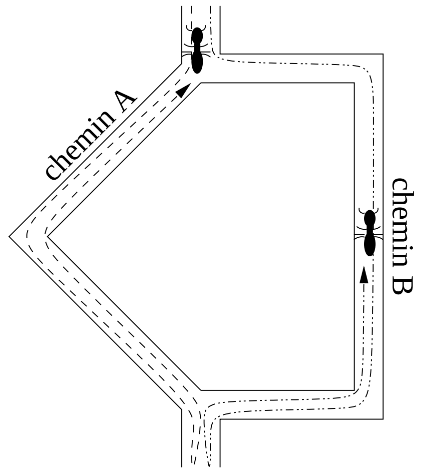


Fig. 1.12 – Premier retour dans un cas où deux fourmis se sont réparties les voies à explorer : le chemin le plus court est renforcé plus vite.

un chemin par une différence de quantité déposée. Mais les grandes lignes du phénomène réel sont conservées : colonie d'individus simples coopérants, stigmergie, localité des comportements...

Ce principe d'optimisation par un comportement collectif a été employé avec succès pour résoudre des problèmes classiques d'optimisation tels que le voyageur de commerce, l'ordonnancement de tâches, le routage dans des réseaux de communication ou la coloration de graphe. Personnellement, j'ai eu l'occasion de le faire appliquer par des élèves sur un problème d'optimisation d'ordre de tests en cas de panne d'un système complexe, les résultats de l'approche s'avérant assez concluants.

Conclusion

Cet exemple, en plus d'illustrer ce qu'est un système multi-agents, met en valeurs la possibilité d'émergence d'un phénomène de groupe intelligent alors que chaque agent n'est ici qu'un automate des plus réactifs. On a clairement un phénomène holiste puisque le tout (qu'est le groupe) est plus que la simple somme de ses parties (que sont les agents). Il n'est pas nécessaire qu'ils adoptent des règles comportementales complexes pour que des agents soient amenés à coopérer efficacement.

Par contre, l'observation de cet exemple amène une question pertinente : comment aurait-on pu en arriver au comportement de ces fourmis en assignant au groupe l'objectif de trouver un plus court chemin ? On voit ici que la conception d'agents coopérants à une tâche donnée n'apparait pas évidente.

1.2.3 Quelques concepts clefs

Si les termes "agent" et "environnement" sont apparus dès le début de notre présentation des systèmes multi-agents, c'est en complétant le vocabulaire lié à ce domaine que nous proposons ici de l'approfondir, en commençant par décrire l'approche des SMA proposée par Demazeau.

Voyelles

Dans un phénomène de comportement collectif tel que celui décrit en section 1.2.2 (recherche d'un plus court chemin par une colonie de fourmis), les agents sont indissociables des relations qui les lient et les font agir de concert. Ces notions duales n'ont aucun sens l'une sans l'autre, de même que nœuds et arcs dans un graphe.

C'est dans une optique comparable qu'a été mise au point l'approche *Voyelles* [Demazeau, 1997] qui considère quatre aspects principaux et complémentaires dans un système multi-agents : les agents, l'environnement, les interactions et l'organisation (d'où les quatre voyelles *AEIO*). En partant de l'idée qu'un SMA est un ensemble d'entités coexistant dans un même univers, essayons de définir ces quatre aspects :

- **Agents** : Les agents sont ces entités autonomes "opérantes". C'est ici l'architecture interne qui nous intéresse dans ce concept déjà défini en section 1.1. Pour revenir à nos fourmis, ce sont elles qui jouent le rôle des agents dans l'exemple sus-cité (attention, ici un automate aussi est un agent).
- **Environnement** : L'environnement est l'ensemble des éléments grâce auxquels les entités (agents) vont pouvoir agir ensemble, être mises en relation. Colonie, nourriture et chemins menant de l'une à l'autre sont par exemple l'environnement des fourmis.
- **Interactions** : Les interactions correspondent aux influences que les actions d'un agent peuvent avoir sur d'autres agents ou sur lui-même. D'une manière assez proche, [Bonabeau et Theraulaz, 1994] définissent une interaction comme *"une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques"*. Chez les fourmis, l'attirance éprouvée par une fourmi pour un chemin et créée par le dépot de phéromones d'une autre foumi est un exemple d'interaction.
- **Organisation** : C'est ce qui va structurer l'ensemble des entités. Des règles sociales peuvent en être à l'origine, définissant des rôles et des contraintes entre ces rôles. Il est à noter que cette organisation peut apparaître et évoluer, comme c'est le cas dans une meute. Dans l'exemple de colonie de fourmis suivi jusqu'à présent, il n'y a pas d'organisation particulière mise en jeu, si ce n'est à travers le chemin suivi par le groupe (pour ce qui est de la recherche de nourriture qui nous préoccupe).[14]

Complément sur l'organisation

Pour donner quelques exemples d'organisations assez typiques, on peut citer : les organisations hiérarchiques [Fox, 1981], structurées en niveaux dans lesquels les décideurs coordonnent les actions des niveaux inférieurs ; les réseaux contractuels [Smith, 1980], dans lesquels une discussion permet de négocier les actions que doit effectuer chacun sous la forme d'un contrat ; ou encore le principe du marché centralisé [Wolisz et Tschammer, 1993] dans lequel un agent courtier sert d'intermédiaire entre acheteurs et centraliseurs.

Ces quelques exemples montrent la variété qui se cache sous ce concept d'organisation. Nous en resterons là sur ce point, en suggérant en complément la lecture de [Malville, 1999] aux lecteurs intéressés par une discussion plus complète de la notion d'organisation dans un système multi-agents.

[14] Je laisse à d'autres le soin de filer une métaphore à l'aide d'araignées sociales (par exemple [Bourjot *et al.*, 2003]).

Complément sur l'interaction

Pour ce qui est de l'interaction entre agents, elle peut être considérée à deux échelles différentes.

A un niveau macroscopique (au niveau global du fonctionnement du SMA), des agents peuvent avoir des buts, des comportements, qui soient :

- *indifférents* (tout à fait indépendants les uns des autres),
- *antagonistes* (agents en conflit sur des ressources, ayant des buts contraires), ou
- *coopérants* (agents dont les objectifs sont communs).

Mais la distinction n'est pas toujours évidente : si l'on ne connait que les règles élémentaires de fonctionnement des fourmis, on risque de les considérer comme indifférentes les unes aux autres, alors qu'émerge un phénomène de coopération à travers les interactions permises par l'utilisation de phéromones. De manière générale, des agents peuvent d'un moment à l'autre se trouver dans des situations aussi bien d'indifférence, d'antagonisme que de coopération.

Ainsi, à une échelle plus réduite, des agents, même globalement indifférents, peuvent avoir localement intérêt à *coordonner* leurs actions pour ne pas nuire à la réussite des objectifs de chacun. Tout le problème des agents va ainsi consister à se mettre d'accord sur les actions que doit effectuer chacun pour que la combinaison de ces actions permette d'atteindre un but commun. Il est à noter que le choix d'une organisation telle que celle de réseau contractuel (citée précédemment) spécifie le moyen qui va rendre la coordination possible (en l'occurence, la négociation).

Retour sur la coopération

La coopération qui vient d'être évoquée s'intéressait à l'adéquation des buts relatifs des agents, adoptant ainsi "le point de vue des agents". Mais ce qui intéresse l'utilisateur d'un SMA, c'est simplement le fait que le comportement du groupe permette d'atteindre un certain objectif. En ce sens, on peut présenter la coopération "du point de vue de l'utilisateur" comme étant le simple fait que le problème posé est bien résolu collectivement. La définition donnée dans [Ferber, 1995] considère qu'il y a coopération si l'une ou l'autre situation qui viennent d'être décrites (les deux points de vue) est vérifiée, créant ainsi une confusion. A l'inverse, [Brassac et Pesty, 1995] ne considère que l'aspect *intentionnel*, c'est-à-dire quand chaque agent veut participer à une tâche collective.

Pour illustrer le fait qu'une coopération "du point de vue de l'utilisateur" n'implique pas nécessairement une coopération "du point de vue des agents", donnons simplement l'exemple de la motivation que peut provoquer la mise en concurrence de sportifs : il semble plus difficile de battre des records quand on court seul que quand on court contre d'autres.

1.2.4 Bilan

Dans cette section 1.2, nous avons essayé de montrer ce que sont les systèmes multi-agents de diverses manières :

- en les distinguant de la résolution distribuée de problèmes d'abord, mais
- mais aussi en les montrant sous la forme de systèmes dynamiques en interaction, et
- enfin à travers l'exemple de fourmis établissant un plus court chemin.

Ce travail a aussi été complété en développant certains concepts-clefs du domaine des SMA tels qu'"interaction", "organisation" et "coopération".

Cet outil de l'intelligence artificielle que sont les SMA permettra de s'attaquer à certains problèmes trop "volumineux" pour une approche monolithique classique (ou pour certains problèmes naturellement posés à un groupe d'agents), et ce, en adoptant une approche holiste, le comportement du SMA allant au-delà qu'une simple somme des capacités des agents qui le constituent.

Les SMA posent malheureusement des problèmes de conception tels que celui de transmettre à tous les agents d'un groupe la notion d'un but commun. Mais nous allons revenir sur ce point dans la section suivante.

1.3 Conception

Ayant présenté les concepts d'agent et de système multi-agents, nous discuterons dans cette dernière section de la *conception* d'agents et de SMA. Notre but avoué est en effet d'automatiser cette conception. Nous allons donc essayer ici de préciser ce que nous entendons par là, diverses approches étant possibles.

1.3.1 Agent seul

En ce qui concerne la conception d'un agent seul, nous allons d'abord voir ici à quoi elle peut s'intéresser (architectures interne et externe). L'illustration de ces deux éléments va naturellement nous amener à l'idée d'une inspiration biologique. Sur la base de ces différents éléments, nous ferons alors un premier point sur l'optique qui sera la notre dans la conception d'un agent.

1.3.1.1 Architecture interne

La description que nous avons faite de ce qu'est un agent le présente comme essentiellement dirigé par une volonté, un but, que l'on peut traduire par une mesure de performance. Vue sous cet angle "système adaptatif", la conception d'un agent consiste à définir cette "fonction objectif" ainsi que l'algorithme qui va s'attacher à la maximiser (si l'on considère un problème de maximisation). Si la mise en pratique le requiert, un certain nombre de connaissances a priori sur le monde peuvent être fournies. Cela va à l'encontre du souhait d'autonomie, mais peut être contraint par le problème traité. Ces trois données sont résumées sur la figure 1.13.

A propos de la mesure de performance, [Russell et Norvig, 1995] fait remarquer qu'elle doit être consciencieusement définie, sous peine d'obtenir un comportement inattendu. C'est ainsi qu'un robot qui aurait pour but d'aspirer un maximum de poussière pourrait en arriver à en re-déposer pour pouvoir la ré-aspirer (ce qui ne lui est pas interdit, voir figure 1.14).[15]

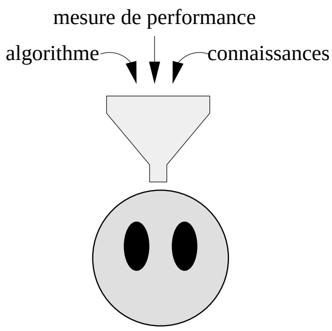


FIG. 1.13 – Les trois points à définir dans un système adaptatif.

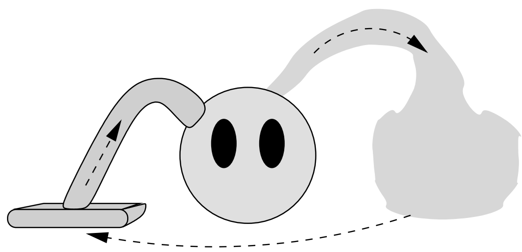


FIG. 1.14 – Comment être un robot-aspirateur efficace.

[15]Un autre exemple, rencontré pendant nos travaux, est évoqué en annexe A.3.1.

1.3.1.2 Architecture externe

Si l'on revient sur une vision un peu plus large de ce qu'est un agent, il faut aussi tenir compte de son architecture. Ainsi, si c'est aussi une décision incombant au concepteur (et non une donnée du problème initial), le choix des capteurs et effecteurs de l'agent va lui aussi être crucial.

Un agent équipé de manière insuffisante par rapport à la tâche qu'il lui faut accomplir se verra dans l'incapacité, quel que soit son comportement, d'agir de manière efficace. A l'inverse, un agent sur-équipé et qui doit s'adapter, c'est-à-dire apprendre, risque d'une part d'être noyé par le flot d'informations venant de ses capteurs, et d'autre part de ne pas savoir quels actions effectuer parmi toutes celles qui lui sont possibles. Entre ces deux situations extrêmes existent des situations intermédiaires qui permettent de faire un compromis entre capacités potentielles de l'agent et complexité de l'architecture à gérer (figure 1.15).

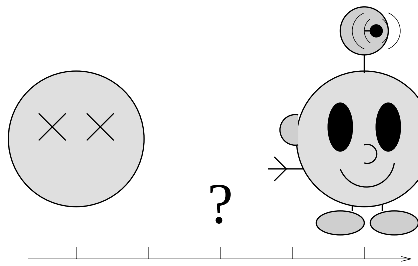


FIG. 1.15 – Agents sous- et sur-équipés pour pousser trouver la sortie d'un labyrinthe (voir et se déplacer sont les seules capacités requises).

1.3.1.3 Problème général

Définitions des architectures interne (fonction objectif, algorithme et connaissances fournies) et externe (capteurs et effecteurs) vont en fait de pair. S'il s'avère nécessaire d'utiliser des observations plus riches, peut-être va-t-il falloir aider l'agent en lui expliquant par exemple comment pré-traiter certaines des informations qu'il reçoit ?

Ce problème se pose pour chaque agent, étant donné son objectif et son environnement. Parlant de la conception d'animats (sujet sur lequel nous allons revenir, mais qui rejoint ici la conception d'agents), [Pfeifer, 2000] explique ainsi qu'il faut "harmoniser" l'environnement avec la morphologie, le matériel et le contrôle d'un animat. On retrouve cette idée dans l'approche de [Wilson, 1991], qui met en valeur l'adéquation entre agent et environnement.

1.3.1.4 Bio-inspiration

S'il est un modèle d'harmonie, c'est bien le travail de dame nature qui, par la sélection naturelle[16], met au point des agents adaptés à leur environnement.

Ainsi, une réponse au problème de conception d'agents peut être l'imitation d'agents naturels, en s'intéressant chez les animaux aux modes de déplacement, aux moyens de manipulation, aux capacités perceptives diverses mises en jeu, mais aussi en essayant de comprendre les mécanismes cognitifs impliqués. Ces deux voies amènent aux deux branches de l'intelligence artificielle qui consistent à imiter la nature dans la classification de [Russell et Norvig, 1995] (comme on l'a évoqué en section 0.2.1).

[16] Autrement dit : par algorithme génétique.

L'approche animat

L'exemple qui semble le plus flagrant d'une telle approche agent "bio-inspirée" est celui de l'approche animat suivie par l'équipe de Jean-Arcady Meyer. L'inspiration biologique est sans équivoque, puisque cette approche s'intéresse à *"la synthèse d'animaux simulés sur ordinateur ou de robots réels dont les lois de fonctionnement sont aussi inspirées que possible de celles des animaux"*. En revanche, le lien avec la notion d'agent est nettement moins évident.

Pour faire ce lien, considérons d'abord l'agent comme une entité qui fait de l'optimisation d'une certaine fonction objectif et ce, à l'aide d'un algorithme donné. De son côté, l'animat est une entité qui mime des mécanismes biologiques. Or les algorithmes tirés des mécanismes biologiques en question sont : des algorithmes génétiques, des réseaux de neurones, des méthodes d'apprentissage... c'est-à-dire des algorithmes d'optimisation, même s'il n'y a pas de réelle optimisation dans le phénomène naturel de départ.

A la lumière de cette comparaison, il me semble qu'un animat puisse être raisonnablement présenté comme un agent conçu par inspiration biologique. Toutefois, s'il doit rester une différence importante, c'est vraisemblablement dans les objectifs suivis par l'une et l'autre approche. Pour avoir un autre point de vue sur l'approche animat, on pourra aussi se référer à la présentation faite dans [Wilson, 1991].

1.3.1.5 Point de vue adopté

Si nous ne refusons pas l'inspiration biologique quand elle semble opportune, ce n'est pas dans cette optique qu'est conduit notre travail (comme nous l'avons explicité en nous positionnant parmi les quatre approches de l'I.A. en section 0.2.1). Il est d'ailleurs à noter que se contenter d'imiter pourrait empêcher certaines innovations[17].

Notre intérêt ne se porte pas non plus sur l'architecture externe de l'agent. C'est néanmoins un travail que nous devons faire puisque nous avons défini nos propres agents "virtuels", et qu'il a fallu les doter de capacités de perception et d'action suffisantes pour accomplir les tâches voulues.

C'est par contre sur la *réalisation de l'architecture interne* (le système chargé de l'adaptation) que nous allons nous pencher. Toute méthode de l'IA étant en fait une manière de réaliser des architectures internes, on ne pouvait se lancer dans un réel état de l'art sur ce point. Dans cette thèse, ce sont bien entendu l'apprentissage par renforcement et les systèmes multi-agents qui seront les deux techniques principales considérées. Toutefois on cherchera à répondre à la difficulté de la tâche d'adaptation par la mise au point de méthodes incrémentales, fondées sur une difficulté croissante du problème posé à l'agent.

1.3.2 Système multi-agents

1.3.2.1 Problème posé

On a vu que, pour un agent seul, une fois les aspects "matériels" réglés (choix de capteurs et effecteurs), on peut voir les aspects logiciels subsistants comme la mise au point d'un algorithme ayant pour fonction de maximiser une mesure de performance.

Pour un groupe d'agents, dont on souhaiterait qu'ils coopèrent, de nouvelles questions se posent. Il ne suffit pas de dire qu'on peut simplement se ramener à la conception de n agents. L'utilisation de méthodes spécifiques va donc s'avérer indispensable.

[17] L'hélice, par exemple, me paraît assez innovante par rapport à ce que sait faire la nature.

1.3.2.2 Bio-inspiration

L'inspiration biologique peut être une première approche, de même que pour la conception d'agents seuls, comme on l'a vu en section 1.3.1.4.

La conception de systèmes multi-agents en s'inspirant de phénomènes biologiques est d'ailleurs un domaine de recherche très important, comme le montrent les nombreux travaux sur l'exemple des colonies de fourmis vu en section 1.2.2 [Bonabeau *et al.*, 1999], sur les araignées sociales [Bourjot *et al.*, 2003] ou encore le comportement social des rats [Thomas *et al.*, 2002]. Ces travaux s'attachent à analyser et comprendre les mécanismes mis en jeu dans des exemples de comportements de groupe émergents, pour essayer soit simplement de les reproduire pour résoudre les problèmes qui s'y prêtent, soit d'en tirer une meilleure compréhension de tels phénomènes et donc une connaissance bien plus "générale".

Néanmoins, nous ne faisons pas ici le choix d'adopter une telle démarche, souhaitant plutôt mettre au point des méthodes de conception directement dirigées par les buts assignés au groupe d'agents.

1.3.2.3 Quelques questions

Revenons donc au problème de départ : obtenir d'agents qu'ils coopérent à une tâche commune. Travailler sur la base d'un objectif (exprimé à travers des récompenses) n'est pas évident. Etant donnée une mesure de la performance du groupe, il est difficile d'en déduire les comportements individuels que les agents doivent adopter pour que, de leur interaction, émerge un comportement de groupe satisfaisant. De fait, les outils théoriques manquent dans ce domaine.

Conception centralisée/décentralisée ?

Une première question est de savoir si l'on veut centraliser la conception (un seul programme monolithique produisant les comportements des différents agents) ou pas. L'intérêt d'un processus centralisé est en général qu'il maîtrise mieux les données du problème, et ainsi peut aboutir à de meilleurs résultats. Il s'agit alors qu'un unique programme optimise $n \times m$ paramètres si l'on manipule un groupe de m agents définis chacun par n paramètres.

Une telle approche amène deux remarques :

1. On doit, avec une conception centralisée, nécessairement distinguer une première phase de pure conception de la phase ultérieure d'utilisation normale du SMA. Mais cette seconde phase n'exclut pas la possibilité que les agents continuent à s'adapter, qu'il s'agisse d'affiner les comportements élaborés ou de répondre à des modifications de l'environnement.
2. Du fait de cette centralisation *et* de ce que la conception requière en général une expérimentation, il faut pouvoir travailler en gardant contact avec tous les agents (puisque ce sont eux qui vivent ces expériences). Une telle situation amène logiquement à considérer le passage par une simulation (donc la nécessité d'un modèle de l'environnement).

L'approche décentralisée, elle, consiste à doter chaque agent de capacités d'adaptations propres, éventuellement en étant capable de "raisonner" sur ce que veulent faire les autres agents (c'est ce qui est fait par exemple à travers la notion d'empathie [Chadès, 2003]) , et de laisser évoluer le groupe. Le problème qui apparaît est que l'amélioration ne se situe plus alors nécessairement au niveau du groupe, mais au niveau de chaque agent. En effet, les mesures de performances accessibles aux différents agents ne refléteront pas de manière sûre les performances du groupe complet (à moins de se placer dans des situations particulières).

Notre préférence ira aux **approches décentralisées**, en sorte que les apprentissages des agents soient indépendants : que chacun apprenne de son côté. Ce choix est principalement guidé par le souhait que les agents soient toujours capables de s'adapter (si certains aspects de l'environnement changeaient par exemple), ne distinguant pas strictement phases de conception et d'apprentissage : on ne cherche pas à obtenir des automates. Toutefois, nous mettrons en œuvre dans nos travaux des outils d'aide à l'apprentissage qui requièrent **une "certaine" centralisation**, ce dont nous rediscuterons le moment venu.

Perception partielle/totale ?

Un cas dans lequel une conception décentralisée, comme nous venons d'en faire le choix, pose de moindres problèmes pour que les différents agents soient informés des performances du groupe est celui d'agents dotés d'une perception totale. Celà ne résoudrait pas pour autant le problème de la coordination des agents puisque, même si chacun sait quelles "actions jointes" du groupe sont optimales, encore faut-il se mettre d'accord sur l'action jointe suivie.

Mais, comme nous l'avons déjà exprimé, les agents sur lesquels nous travaillons n'ont qu'une **perception partielle** de leur environnement. Ce choix est lié d'une part au souhait d'être réaliste en ayant des agents qui souffrent d'un manque d'informations, et d'autre part à la simplification de la tâche qui peut en découler, puisque l'espace de recherche en sera considérablement réduit.

Distinction des autres agents ?

Un dernier point que nous évoquerons ici n'est en fait qu'un des aspects des capacités perceptives restreintes des agents. Il s'agit de savoir si un agent A est capable de distinguer d'autres agents B et C l'un de l'autre. Cette question a son importance puisqu'une telle incapacité a l'avantage de simplifier les observations de l'agent, mais nuit aux possibilités de spécialisation des agents. Cette forme de perception partielle (où l'on ne sait pas distinguer deux agents l'un de l'autre) est intéressante parce que particulièrement liée aux capacités sociales des agents.

Deux exemples simples mettant en jeu nos trois agents A, B et C vont permettre de montrer le désavantage que constitue une non-différenciation d'autrui (figure1.16). Dans les deux cas, les agents ont pour soucis de savoir se croiser dans un monde plan. Les deux problèmes se différencient par l'orientation des mondes plans dans lesquels ils se situent :

- Dans le cas (a), le plan est horizontal. Les agents peuvent alors apprendre, par exemple, à se contourner par la droite.
- Dans le cas (b), le plan est en revanche vertical. Dans cette situation, si A voulait passer au-dessus de tout autre agent et donc B au-dessous (ne sachant pas distinguer autrui, un comportement déterministe suivrait de telles règles), C ne saurait quelle attitude adopter face à un agent qu'il n'identifierait pas.

Note : Au moins 3 agents sont nécessaires pour qu'existe ce problème de coordination.

Par la suite, et malgré les inconvénients que cela apporte, nous supposerons que les agents **ne savent pas se distinguer** les uns des autres, à moins qu'ils ne soient explicitement de "types" différents. Ce choix vient d'une part du fait qu'une telle capacité de distinction est assez complexe pour un agent, et d'autre part cela ferait croître de manière inconsidérée le nombre de situations possibles aux yeux d'un agent (alors que ce nombre est déjà naturellement grand).

1.3.2.4 Point de vue adopté

Cette section 1.3.2 a rapidement discuter de quelques points de la conception de systèmes multi-agents. Il ne s'agissait que de préciser certains choix dans notre approche, et non de réel-

a) Plan horizontal : pas de problème.

b) Plan vertical : pas de comportement déterministe optimal.

FIG. 1.16 – Problème de croisement d'agents qui ne se différencient pas dans un monde plan.

lement s'attaquer aux difficultés de conception d'agents coopérants (ce qui viendra au chapitre 3).

Pour récapituler, notons d'abord que ce qui a été dit sur notre point de vue au niveau mono-agent (en section 1.3.1.5) reste ici valable. L'aspect incrémental qui apparaîtra dans nos approches jouera d'ailleurs un rôle aussi bien à l'échelle d'un agent qu'à l'échelle du groupe. Pour ce qui est des spécificités du cadre multi-agents, elles se résument par le choix d'une conception "semi-centralisée", de perceptions partielles et d'une incapacité à distinguer les autres agents entre eux.

1.4 Bilan

Nous nous sommes attachés dans ce premier chapitre à aborder le monde des agents à la fois pour développer notre vision de l'intelligence artificielle, laquelle passe par le point de vue "agent", et pour faire une introduction au domaine des systèmes multi-agents, en montrant ce qu'ils permettent et les problèmes que leur utilisation soulève.

Ce chapitre peut paraître assez hétéroclite, du fait qu'il est l'occasion à la fois de présenter tout un ensemble de concepts dont les définitions ne font pas l'unanimité, et d'exposer pour partie l'optique dans laquelle nous travaillons. Nous reviendrons sur certains aspects et préciserons certains points le moment venu dans la suite de ce mémoire.

Le chapitre suivant va revenir au point de vue d'un agent seul, de manière à présenter des outils d'apprentissage par renforcement répondant aux souhaits que nous avons exprimés quant à ce que doit être un agent.

Chapitre 2

Apprentissage par Renforcement

Sommaire

Ce deuxième chapitre va débuter par une section 2.1 dont l'objectif est de présenter les origines de l'*apprentissage par renforcement* (A/R) en faisant d'une part le lien avec la notion d'agent vue au chapitre précédent, et d'autre part en justifiant le choix de modèle d'apprentissage par renforcement qui a été fait : celui des *processus de décision markoviens* (PDM) et de leurs dérivés. La section 2.2 s'efforcera de faire une présentation de ce formalisme, en s'intéressant plus particulièrement à l'apprentissage en-ligne par opposition à la conception de plans réactifs hors-ligne, et en concluant sur les limitations de cet outil. Ceci conduira à voir en section 2.3 ce que sont les *processus de décision non-markoviens* (PDNM) et les approches de résolution de ces problèmes, en développant plus particulièrement les algorithmes qui seront utilisés dans la suite du mémoire.

2.1 Origines de l'apprentissage par renforcement

2.1.1 Point de départ

Un agent a été présenté au chapitre 1 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. Il s'agira en général d'apprendre un *comportement maximisant une mesure de performance*, à moins que cet objectif soit seulement lié à la compréhension de l'environnement.

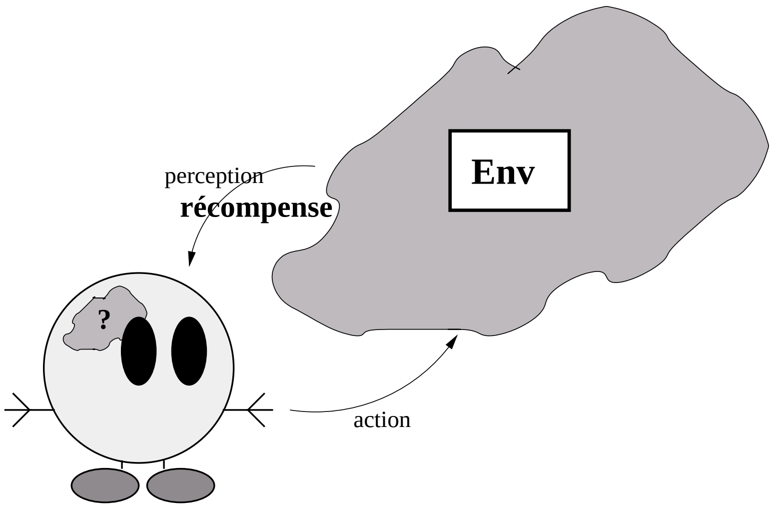


FIG. 2.1 – Boucle (perception+récompense)/action en apprentissage par renforcement : la mesure de performance du comportement de l'agent est la *moyenne temporelle de ses gains.*

Cette mesure peut se déduire d'un signal de renforcement, récompense évaluée de manière numérique (éventuellement négative) et que l'agent cherche à optimiser à long terme. Cette récompense peut être perçue comme venant de l'environnement (comme le présente la figure 2.1) ou comme un jugement interne propre à l'agent. Ce deuxième point de vue aura notre préférence, puisqu'il correspond à l'idée qu'un point essentiel dans la conception d'un agent est la définition de ses objectifs, ici traduits par sa *fonction de récompense.*

Ayant donc choisi de concevoir des agents par l'intermédiaire de méthodes d'apprentissage par renforcement (voir section 0.3.1), un certain nombre d'outils peuvent être employés pour traduire ce paradigme.

2.1.2 Modèles biologiques

Encore une fois, des travaux de modélisation biologique peuvent servir de base pour développer un formalisme d'apprentissage par renforcement.

Un exemple type est celui du modèle psychologique du conditionnement classique (pavlovien) dû à [Rescorla et Wagner, 1972]. Ce modèle décrit de manière simple le renforcement de la valeur associée à un stimulus, lorsque celui-ci apparaît. Il est par contre insuffisant pour en arriver à améliorer un comportement complet.

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, 1982]. On dispose alors d'un outil qui se base sur des traîtements numériques, ce qui répond au caractère situé de l'agent et permet à la fois de gérer l'incertitude et d'éviter l'emploi de symboles. Cette approche efficace a conduit à de nombreux développements dans le domaine du connexionnisme.

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 algorithmes. Cherchant un cadre de travail qui offre de telles possibilités, on préfèrera l'utilisation de modèles qui soient plus spécifiquement adaptés à de tels objectifs, et dont il va être maintenant question.

2.1.3 Outils mathématiques

On n'essaiera pas ici de reconstruire un historique complet des avancées dans le domaine des mathématiques qui ont amené aux outils récents les plus efficaces pour travailler sur l'apprentissage par renforcement. Les liens et influences sont trop complexes pour être bien connus et représentés, et il faudrait revenir à la découverte du concept de nombre pour ne rien oublier.

On se contentera de citer :

- d'abord la *théorie des probabilités*, outil capable de représenter et gérer l'incertitude ;
- puis les *chaînes de Markov*, formalisme (au sein de la théorie des probabilités) permettant l'étude d'un système dont l'évolution est décrite par des lois stochastiques ;
- enfin 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 ;
- et plus précisément la *théorie de l'utilité*, parce qu'elle apporte la notion d'*évaluation* d'une décision, notion que l'on va pouvoir mettre en correspondance avec la notion de performance.

Comme l'introduction de [Horvitz *et al.*, 1988] le met en valeur, ces différentes théories regroupent les différents aspects utiles à une formalisation de l'apprentissage par renforcement. Il reste à voir comment les unifier précisément sur ce problème.

2.1.4 Principe

Avant d'entrer dans les détails des processus de décision markoviens dans la section 2.2 qui suit, commençons par tenter d'expliquer leur principe sans formules. Cette explication se fonde sur les différents domaines évoqués ci-dessus.

Le point de départ est l'utilisation de chaînes de Markov. Comme on l'a évoqué, cet outil de la théorie des probabilités permet d'étudier l'évolution des états d'un système (existence d'un régime stationnaire, probabilité de présence en un état...) en connaissant les lois qui régissent les transitions du système, en l'occurence la probabilité de passer d'un état s à un autre s' (les états précédents n'ayant pas d'influence).

L'idée des processus de décision markoviens est alors de faire un lien entre chaînes de Markov et théorie de l'utilité. En effet, il s'agit d'une part de définir des valeurs estimant la “qualité” d'un état ou d'une transition, et d'autre part de permettre un contrôle partiel sur les transitions (grâce à ce qu'on appellera des *actions*).

De la simple étude de l'évolution d'un système, on passe ainsi à la possibilité de le contrôler, en ayant en plus une évaluation de sa qualité.

2.1.5 Point de vue

Dans la présentation des processus de décision markoviens (et de certains dérivés) qui va suivre, nous limiterons notre propos de manière à aboutir à des formalismes et algorithmes correspondants au type d'agent que nous cherchons à concevoir (voir section 1.1.6) : sans mémoire à court terme ni modèle de son environnement, avec des perceptions partielles, et sans moyens de communication (pour rappeler certains points essentiels).

Avant de travailler en respectant toutes ces conditions, nous commencerons par un cadre aux hypothèses assez fortes, celui des processus de décision markoviens, pour relâcher ensuite progressivement ces hypothèses.

2.2 Processus de Décision Markoviens

Cette section d'introduction aux PDM s'organise simplement en une description du formalisme lui-même (2.2.1), puis une présentation de méthodes classiques de résolution (2.2.2) et enfin une explication des limites de cet outil (2.2.3), lesquelles vont nous contraindre à explorer d'autres horizons.

Note : Le lecteur intéressé par une présentation plus complète pourra se référer par exemple à [Puterman, 1994] ou [Sutton et Barto, 1998], ou encore [Fabiani *et al.*, 2001] (document en langue française).

2.2.1 Formalisme

La présentation du formalisme PDM faite ici commence par fournir les données du problème qu'est un PDM, pour ensuite expliquer plus précisément son fonctionnement et introduire d'autres notations utiles à sa compréhension comme aux outils de résolution qui l'accompagnent.

2.2.1.1 Définition $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$

Un *processus de décision markovien* est défini par un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ dans lequel :

- $\mathcal{S} = \{s\}$ est un ensemble d'*états*,
- $\mathcal{A} = \{a\}$ est un ensemble d'*actions*,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0;1]$ est une *fonction de transition*. Elle donne la probabilité de passer entre les instants t et $t+1$ de l'état s à l'état s' sachant l'action choisie a comme suit :

$$\mathcal{T}(s, a, s') = P(s_{t+1} = s' \mid s_t = s, a_t = a).$$

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Pi(\mathbb{R})$ une *fonction de récompense*[18] qui, à une transition, associe une distribution de probabilité sur $\mathbb{R}$. En général, on utilisera simplement la valeur moyenne de cette distribution, en gardant la notation $\mathcal{R}(s, a, s')$.

[18] On parlera aussi de renforcement ou de gain.

On se restreindra à des PDM :

- finis (dont les ensembles $\mathcal{S}$ et $\mathcal{A}$ sont finis),
- à horizon infini (sans limite temporelle ni état mettant fin à l'évolution du système), et
- homogènes ($\mathcal{T}$ et $\mathcal{R}$ sont indépendants du temps).

Un PDM définit simplement un problème de prise de décision. La mesure de performance considérée est a priori la moyenne temporelle des récompenses.

Exemple

On peut illustrer ce formalisme sur un problème de recherche de plus court chemin dans un labyrinthe tel que celui de la figure 2.2. Les états sont alors les positions possibles de l'agent dans ce labyrinthe, et les actions correspondent aux quatre déplacements dans les directions Nord, Sud, Est et Ouest. Il est à noter qu'ici l'agent n'est pas réellement perdu, puisqu'il connaît sa position actuelle.

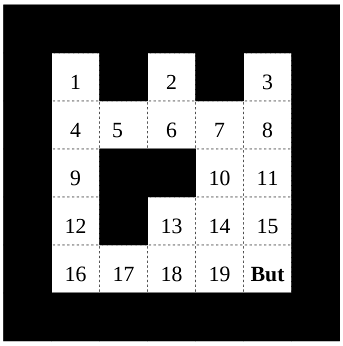


FIG. 2.2 – Un labyrinthe simple (avec sa case **B**ut) à 20 états : les 20 cases blanches.

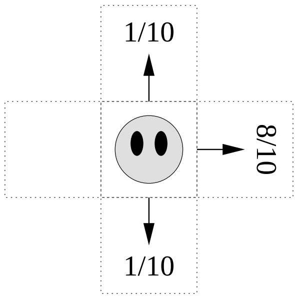


FIG. 2.3 – L'effet de l'action "Est" en l'absence d'obstacles (au bout des flèches sont indiquées les probabilités de chaque mouvement).

Si les mouvements de l'agent sont bruités, la fonction de transition est non déterministe (évolution incertaine). Cela va se traduire dans le cas présent par un agent qui va 8 fois sur 10 dans la direction souhaitée, et le reste du temps dans les deux directions voisines, comme l'illustre la figure 2.3. Evidemment, si dans la direction ainsi déterminée se trouve un obstacle, l'agent ne bouge pas.

Il ne reste qu'à spécifier l'objectif de l'agent, une simple case but à atteindre en l'occurence, ce qui amène à définir la fonction de récompense comme étant nulle partout, sauf quand une transition amène sur la dite case, auquel cas elle vaut 1.

Note : Pour que ce problème soit à horizon infini, on peut adapter la loi de transition de manière à ce que, dès qu'il arrive à son but, l'agent soit "télétransporté" au hasard quelque part dans le labyrinthe. De cette nouvelle position, il pourra recommencer sa recherche de la case but.

2.2.1.2 Propriété de Markov

La propriété de Markov est un point essentiel vérifié par tout PDM tel que défini précédemment. Cette propriété dit simplement que la connaissance de l'*état courant* est suffisante pour savoir comment va évoluer le système. Aucune information supplémentaire ne permettrait de mieux prédire le prochain état rencontré (étant donnée l'action choisie).

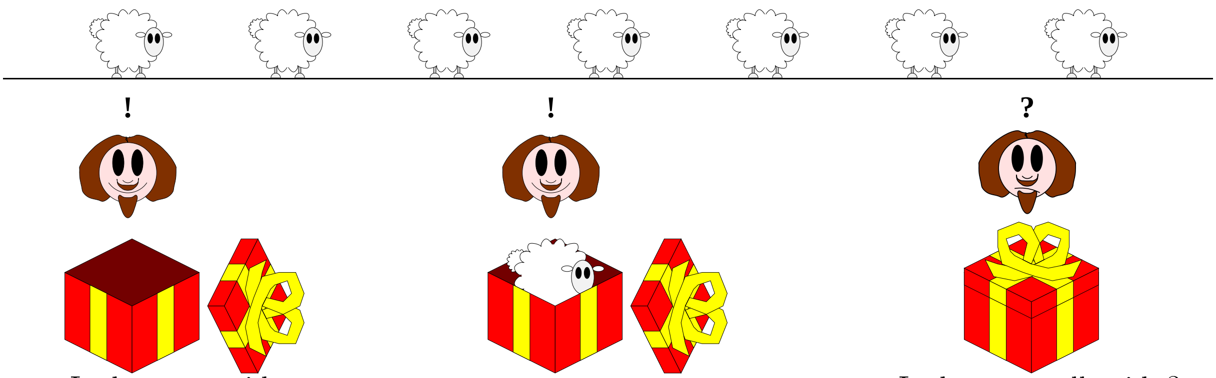
L'historique des états et actions passées étant la seule source d'autres informations, cette propriété va se traduire par la formule suivante :

$$\forall s' \in \mathcal{S} \; P(s_{t+1} = s' \mid s_t, a_t) \quad = \quad P(s_{t+1} = s' \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_1, a_1, s_0, a_0). \qquad (2.1)$$

Exemple

Pour illustrer cette propriété, nous reprenons un exemple de [Dutech et Samuelides, 2003] où elle n'est justement pas vérifiée (voir aussi figure 2.4) :

> « Supposons que nous voulons faire un paquet cadeau. La démarche est simple : ouvrir la boîte, mettre le cadeau dans la boîte, fermer la boîte, emballer la boîte dans du papier cadeau. Si nous ne pouvons observer que l'état de la boîte (*fermée*, *vide*, *pleineEtOuverte* ou *emballée*), nous sommes face à un PDMPO[19]. En effet, quand la boîte est fermée, nous ne pouvons plus savoir si le cadeau est à l'intérieur ou non, sauf si nous avons une mémoire du passé. C'est seulement dans ce cas que nous pouvons savoir si nous avons mis le cadeau à l'intérieur précédemment. Par contre, quand la boîte est ouverte, il n'y a pas d'ambiguïté sur l'état de l'environnement et nous pouvons choisir l'action optimale en fonction de la présence ou non du cadeau dans la boîte [...]»



a- La boîte est vide : on la remplit.

b- La boîte est pleine : on la referme.

c- La boîte est-elle vide ? Faut-il l'ouvrir ou l'envoyer ?

FIG. 2.4 – Un problème de décision partiellement observable : une mémoire aiderait à la prise de décision.

En ce qui nous concerne, ce n'est qu'à partir de la section 2.3 que des cas ne vérifiant pas la propriété de Markov seront abordés. Pour l'instant, les informations immédiates fournies par les perceptions seront suffisantes pour prendre des décisions optimales.

2.2.1.3 Politique

Le formalisme décrit en section 2.2.1.1 permet de poser un problème d'apprentissage par renforcement à travers le fonctionnement du système (T), les moyens d'interaction de l'agent avec l'environnement ($\mathcal{S}$ et $\mathcal{A}$), et le signal de renforcement $\mathcal{R}$ que l'on cherche à optimiser. Il faut alors trouver un comportement optimal.

Une solution (pas nécessairement optimale) à un tel problème est une application qui, à un état $s \in \mathcal{S}$ fait correspondre une distribution de probabilités sur les actions possibles $\Pi(\mathcal{A})$. On

[19] Processus de décision markovien partiellement observable, dont nous reparlerons en section 2.3.

parlera de la *politique* de l'agent, que l'on notera π. Le fait qu'on ne tienne compte que de l'état courant est lié à la propriété de Markov, qui assure que toute l'information nécessaire à la prise de décision est dans l'état courant.

Chaîne de Markov

L'application d'une politique π donnée à un PDM permet de définir une chaîne de Markov. C'est immédiat en utilisant une représentation matricielle de $\mathcal{T}$ et π. L'évolution de la chaîne de Markov obtenue est alors décrite par $\mathcal{T} * \pi$.

2.2.1.4 Mesure de performance

Il a été question jusqu'à présent d'optimiser la récompense de l'agent sur le long terme. Cela a été traduit par le souhait de maximiser le gain moyen futur, qui peut s'écrire :

$$\lim_{N \to \infty} E\left(\frac{1}{N}\sum_{t=0}^{N} r_t\right) \tag{2.2}$$

Mais dans les faits, différentes mesures de performance peuvent être rencontrées (comme ce sera notre cas par la suite). Ce sont toutes des critères basés sur la fonction de récompense $\mathcal{R}$, mais ayant des interprétations et des conditions d'utilisation diverses.

Un premier aspect que l'on peut évoquer est que, si un critère amène effectivement à maximiser le gain moyen à long terme, ce n'est pas pour autant que l'on va arriver au plus vite à un "régime de croisière" optimal. Il sera souvent intéressant de réussir à optimiser aussi bien le résultat futur que le temps mis pour l'atteindre.

Nous verrons plus tard d'autres critères qui, s'ils ont le défaut de favoriser un peu le gain à court terme, s'avèrent très pratiques pour résoudre des PDMs par des méthodes dont la convergence vers un optimum *global* est prouvée. Pour l'instant, l'important est de retenir qu'il n'existe pas qu'un seul moyen de mesurer la performance d'un agent.

2.2.1.5 Mais que sait l'agent ?

On a jusqu'ici expliqué qu'un PDM décrit un problème d'apprentissage par essais-erreurs, qu'une solution se présente sous la forme d'une fonction de décision appelée politique et que la mesure de performance que l'on cherche à optimiser peut prendre différentes formes, mais il n'a pas été dit ce que l'agent est supposé connaître du problème : le tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ ne représente pas le point de vue de l'agent mais le système agent+environnement complet.

Si l'agent connaît toujours les états et actions du système, il n'en sait pas nécessairement autant du fonctionnement de l'environnement ($\mathcal{T}$) et des situations provoquant des récompenses ($\mathcal{R}$). Bien que ces deux grandeurs ne soient pas nécessairement liées, on distinguera en général deux situations d'apprentissage [Kaelbling *et al.*, 1996] :

- **avec modèle** : L'agent connaît $\mathcal{T}$ et $\mathcal{R}$, ou les estime, et l'apprentissage utilise ces modèles.
- **sans modèle** : L'agent cherche à apprendre directement une politique, sans passer par $\mathcal{T}$ et $\mathcal{R}$. Il n'y a pas alors usage de modèles explicites tels que décrits dans le formalisme PDM.

Dans le second cas, il faut nécessairement que l'agent expérimente, qu'il apprenne par essais-erreurs. Le premier cas peut, pour sa part, aussi bien fonctionner sur la base de l'expérimentation que sur une approche de planification (préparation hors-ligne d'un plan réactif). *Nos travaux ne supposeront pas la connaissance d'un modèle*, c'est donc vers les algorithmes qui répondent à cette situation que la suite de cette présentation va s'orienter.

2.2.2 Méthodes classiques

La présente section va s'intéresser aux algorithmes qui ont fait le succès des processus de décision markoviens, entre autres en raison des preuves de convergence vers un optimum global qui les accompagne. Ces algorithmes se fondent sur une utilisation de la programmation dynamique rendue possible par une mesure de performance spécifiquement définie.

2.2.2.1 Principe de Bellman et PDM (fonctions de valeur)

Principe de Bellman

Un principe souvent utilisé pour la résolution de problèmes d'optimisation à plusieurs étapes de décision est celui énoncé dans [Bellman, 1957]. Il dit qu'*une suite optimale de commandes [pour la résolution de tels problèmes] a la propriété que, quels que soient l'étape, l'état et la commande initiaux, les commandes suivantes doivent constituer une suite optimale de décisions pour la suite du problème, en considérant comme conditions initiales l'étape et l'état résultant des précédentes commandes.*

En d'autres termes, si l'on sait trouver la solution optimale (suite de décisions) à partir de l'étape $t+1$ quel que soit l'état s_{t+1}, alors on peut trouver la décision optimale à l'étape t pour tout état possible s_t (et donc travailler par récurrence). Notons $V_t(s_t)$ l'espérance de gain maximale à partir de l'état s_t. Dans cette récurrence qui fonctionne du futur vers le passé, on choisit pour l'état s_t la décision (l'action) a qui maximise l'espérance de gain $E_{s_{t+1}}(R(s_t, a, s_{t+1}) + V_{t+1}(s_{t+1}))$.

Sur l'exemple de la figure 2.5, de l'état s_t, l'action a permet ici de passer aux états s^i_{t+1} avec $i \in \{1..5\}$, avec des probabilités $T(s_t, a, s^i_{t+1})$ et récompenses $R(s_t, a, s^i_{t+1})$. Depuis chaque s^i_{t+1}, le futur permet d'espérer $V_{t+1}(s^i_{t+1})$. L'espérance de gain pour cette paire état-action (s_t, a), et donc pour cet état (en optimisant la décision), valent ainsi :

$$
\begin{aligned}
Q_t(s_t, a) &= \sum_{i=1}^{5} T(s_t, a, s^i_{t+1}) * \left[R(s_t, a, s^i_{t+1}) + V_{t+1}(s^i_{t+1})\right], && (2.3)\\
\text{et } V_t(s_t) &= \max_{a \in \mathcal{A}} Q_t(s_t, a). && (2.4)
\end{aligned}
$$

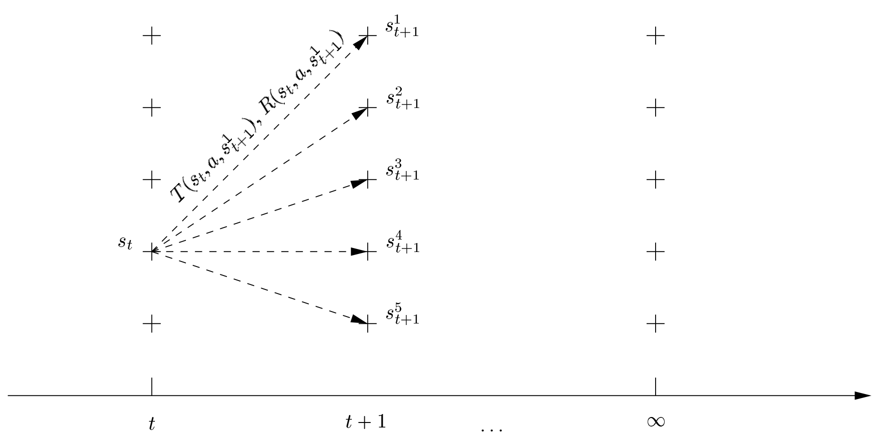


FIG. 2.5 – Comment fonctionne le principe de Bellman.

Application aux PDM

Les notations qui viennent d'être employées ont été choisies par analogie avec les processus de décision markoviens. Toutefois, un problème sérieux se pose pour pouvoir appliquer le principe de Bellman à l'optimisation de politique pour un PDM : le fait que l'on travaille ici à horizon temporel infini.

Travaillant avec cet horizon temporel infini, on va chercher à s'affranchir du paramètre t. Mais ré-écrire les équations 2.3 et 2.4 en ôtant tout simplement le paramètre temps t amènerait généralement à une divergence des calculs.

Par contre, en introduisant un facteur γ de décompte ($\gamma \in]0,1[$), on obtient un système d'équations que l'on écrira encore en deux étapes comme suit :

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A},\ Q(s,a) = \sum_{s' \in \mathcal{S}} T(s,a,s') * \left[R(s,a,s') + \gamma * V(s'))\right], \qquad (2.5)$$

$$\text{et } \forall s \in \mathcal{S},\ V(s) = \max_{a \in \mathcal{A}} Q(s,a). \qquad (2.6)$$

Ce système admet une unique solution notée V^* [Bellman, 1957] (à laquelle est associée une unique fonction Q^*), dont nous allons voir maintenant la principale propriété.

Optimalité

Définissons, pour une politique stochastique π donnée (γ étant fixé), la *fonction d'utilité* V_π par :

$$\forall s \in \mathcal{S},\ V_\pi(s) = E_\pi\left(\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1}) \mid s_0 = s\right). \qquad (2.7)$$

Cette fonction d'utilité permet d'évaluer l'espérance de gain décompté en supposant que l'on suit la politique π à partir de l'état s.

A priori, il peut très bien exister deux politiques π_1 et π_2 et deux états s_1 et s_2 tels que :

$$V_{\pi_1}(s_1) > V_{\pi_2}(s_1) \text{ et}$$
$$V_{\pi_1}(s_2) < V_{\pi_2}(s_2).$$

Il ne s'agit donc pas au premier abord d'un outil de mesure de la performance d'une politique. V ne définit pas une relation d'ordre sur les politiques. Pourtant, on peut montrer qu'il existe au moins une politique π^* dont l'espérance de gain décompté dans n'importe quel état est au moins aussi bonne que celle de tout autre politique :

$$\forall \pi,\ \forall s \in \mathcal{S},\ V_{\pi^*}(s) \geq V_\pi(s). \qquad (2.8)$$

Il existe donc, au sens de cette fonction d'utilité, des politiques qui soient meilleures que toutes les autres, puisque meilleures en tout point. Toutes ces politiques “optimales” ont logiquement la même utilité, qui se trouve être la solution V^* des équations 2.5 et 2.6 [Bellman, 1957]. On définira par ailleurs Q^* la table de Q-valeurs associée à V^*.

On aboutit bien ainsi à un lien entre le principe de Bellman et une mesure de performance d'une politique. Et à supposer que l'on connaisse Q^*, on peut en déduire une politique optimale π^* (qui se trouve être déterministe) par :

$$\forall s \in \mathcal{S},\ \pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a) \qquad (2.9)$$

2.2.2.2 Un algorithme de planification : *value iteration*

Contraction

Les résultats théoriques présentés jusqu'à maintenant supposent qu'un modèle ($\mathcal{T}$ et $\mathcal{R}$) est connu. Nous allons rester dans ce cadre pour voir une première façon d'évaluer V^* (et donc Q^*), puisque c'est ce problème que permet de résoudre l'algorithme *value iteration*.

Cet algorithme se base sur le fait que le système d'équations 2.5 et 2.6 définit une contraction. En effet, en les ré-écrivant comme une unique formule de récurrence, on peut définir une suite de fonctions V_t par :

$$\begin{aligned} \forall s \in \mathcal{S},\ V_0(s) &= \text{valeur quelconque, et} && (2.10)\\ \forall t \geq 0,\ \forall s \in \mathcal{S},\ V_{t+1}(s) &= \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s,a,s') * \left[R(s,a,s') + \gamma * V_t(s')\right]. && (2.11) \end{aligned}$$

La contraction de ce processus itératif provient du fait que, pour tout $t \geq 1$, on a :

$$\max_{s \in \mathcal{S}} |V_{t+1}(s) - V_t(s)| \quad \leq \quad \gamma * \max_{s \in \mathcal{S}} |V_t(s) - V_{t-1}(s)| \qquad (2.12)$$

Du fait de la contraction, cette suite converge vers le point fixe V^* cherché.

Critère d'arrêt

On peut définir un critère d'arrêt selon la précision avec laquelle on souhaite connaître V^* en déduisant de 2.12 (voir [Williams et Baird, 1993]) que :

$$\max_{s \in \mathcal{S}} |V^*(s) - V_t(s)| \quad \leq \quad \frac{\gamma}{1-\gamma} * \max_{s \in \mathcal{S}} |V_t(s) - V_{t-1}(s)| \qquad (2.13)$$

Etant donné $\epsilon > 0$, si l'on s'arrête quand $\max_{s \in \mathcal{S}} |V_t(s) - V_{t-1}(s)| < \epsilon$, alors on sait que V^* est approché par V_t :

$$\max_{s \in \mathcal{S}} |V^*(s) - V_t(s)| \quad < \quad \frac{\gamma}{1-\gamma} * \epsilon \qquad (2.14)$$

Algorithme

De ce calcul itératif (2.10 et 2.11) et du critère d'arrêt qui vient d'être vu, on peut déduire l'écriture de *value iteration* présentée par l'algorithme 1.

Comme on l'a dit, on peut alors obtenir une estimation de Q^* et en déduire une politique déterministe optimale.

Exemple

Nous proposons de montrer à travers un exemple l'évolution de la fonction de valeur approchée en employant une version modifiée du problème décrit en section 2.2.1.1. Il ne s'agit que d'illustrer le fonctionnement de l'algorithme puisque, pour simplifier le problème, on va supposer (contrairement à nos hypothèses de départ) :
- qu'aucune incertitude n'est introduite dans les actions, et
- que la récompense 1 est donnée quand on arrive dans l'état but, lequel est aussi un état final (le système s'arrête quand l'état est atteint).

L'évolution de la fonction de valeur, initialement nulle, est donnée par la figure 2.6. On voit clairement la "vague" s'étendre progressivement dans le labyrinthe.

Algorithme 1 Value iteration

Entrées: Un PDM $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ et un coefficient $\gamma \in [0, 1]$

Initialiser V_0 arbitrairement

Initialiser $t = 0$

répéter

  $t \leftarrow t + 1$

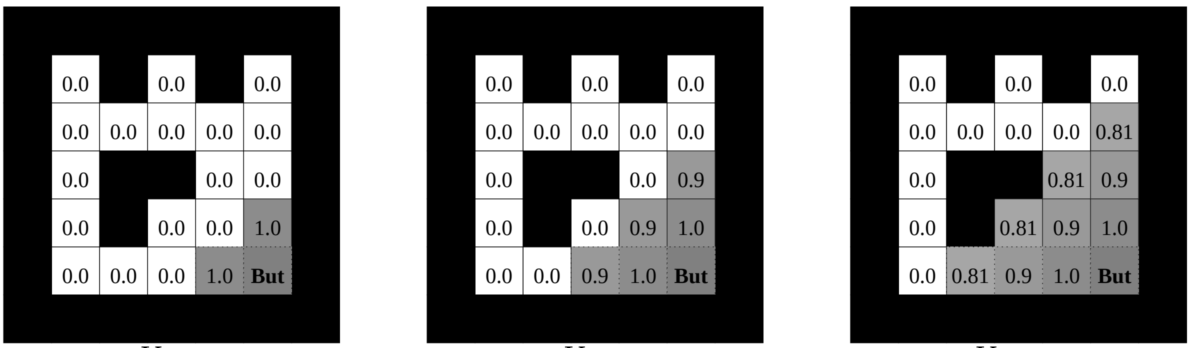
  pour tout $s \in \mathcal{S}$ **faire**

$$V_{t+1}(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} T(s, a, s') * \left[R(s, a, s') + \gamma * V_t(s')) \right]$$

  fin pour

jusqu'à $\max_{s \in \mathcal{S}} |V_t(s) - V_{t-1}(s)| < \epsilon$

Sorties: V_{t+1} approximation de V^* à $\frac{\gamma}{1-\gamma} * \epsilon$ près



$V_{t=1}$, $V_{t=2}$, $V_{t=3}, \ldots$

FIG. 2.6 – Evolution de la fonction de valeur approchée dans le problème du labyrinthe sans incertitude.

2.2.2.3 Q-learning

Comme écrit en section 2.2.1.5, c'est vers des approches d'apprentissage sans modèle que nous nous orientons, alors que le principe de l'algorithme vu précédemment suppose $\mathcal{T}$ et $\mathcal{R}$ connus (ou appris par l'expérience). Le Q-learning [Watkins, 1989] qui va être ici présenté est une adaptation de value iteration qui remplace la mise à jour utilisant $\mathcal{T}$ et $\mathcal{R}$ par une mise à jour en fonction d'une nouvelle expérience.

Une première remarque est que l'algorithme 1 présenté en utilisant la grandeur V peut être transcrit sans difficulté pour travailler sur la grandeur Q avec la formule suivante :

$$Q_{t+1}(s,a) \quad \leftarrow \quad \sum_{s' \in \mathcal{S}} T(s,a,s') * [R(s,a,s') + \gamma * \max_{a' \in \mathcal{A}} Q_t(s',a')] \tag{2.15}$$

Comme c'est de Q que l'on peut déduire la politique déterministe à adopter, c'est directement cette grandeur (que l'on appellera parfois *qualité* d'une paire état-action) que l'on va apprendre. Le principe est que, pour toute nouvelle expérience $(s,a) \rightarrow (s',r)$, la nouvelle estimation de $Q(s,a)$ se partage entre l'ancienne estimation (que l'on notera aussi $Q(s,a)$) et la nouvelle expérience, laquelle laisse espérer un gain futur $r + \gamma * \max_{a' \in \mathcal{A}} Q(s',a')$.

L'importance relative entre les deux termes (estimation précédente et nouvelle expérience) est réglée par un coefficient α_t, lequel doit évoluer pour que l'on tienne de moins en moins compte des nouvelles expériences. Si elles sont importantes au début de l'apprentissage, leur poids doit s'amoindrir au fur et à mesure que l'évaluation de Q se précise. La mise à jour à l'aide d'une nouvelle expérience $(s,a) \rightarrow (s',r)$ s'écrit donc :

$$Q(s,a) \quad \leftarrow \quad \text{estimation précédente} \; + \; \text{nouvelle expérience} \tag{2.16}$$

$$\leftarrow \quad (1-\alpha_t) * Q(s,a) + \alpha_t * [r + \gamma \max_{a'} Q(s',a')] \tag{2.17}$$

Algorithme théorique

De là, on va déduire un algorithme théorique et ses versions pratiques. En effet on peut prouver que, sous certaines conditions peu réalistes, l'algorithme du Q-learning converge presque sûrement vers la solution optimale Q^*. Ces conditions sont, d'après [Jaakkola *et al.*, 1994a] [20] :

- que la trajectoire (s_t, a_t) soit récurrente dans $\mathcal{S} \times \mathcal{A}$ (que l'on passe indéfiniment par tous les couples état-action),
- que $\mathcal{S}$ et $\mathcal{A}$ soient finis (hypothèse qui correspond à l'un de nos choix), et
- que (α_t) satisfasse les hypothèses ordinaires de l'approximation stochastique :

$$\sum_{t=0}^{\infty} \alpha_t = \infty \quad \text{et} \quad \sum_{t=0}^{\infty} {\alpha_t}^2 < \infty \tag{2.18}$$

On obtient alors l'algorithme 2.

Algorithme pratique

La version théorique du Q-learning n'est pas très réaliste puisqu'elle ne s'intéresse qu'à la recherche de Q^*, sans se préoccuper d'utiliser les connaissances acquises. Comme le discute [Thrun, 1992], en pratique, il est souvent bon d'améliorer l'un ou l'autre des deux aspects suivants (voire les deux) :

[20] [Bertsekas et Tsitsiklis, 1989] et [Watkins et Dayan, 1992] montrent la convergence sous des hypothèses légèrement différentes

Algorithme 2 Q-learning (algorithme théorique)

$Q(s,a) \in \mathbb{R}$ arbitrairement initialisé pour tout $(s,a) \in \mathcal{S} \times \mathcal{A}$
Initialiser s
pour tout pas de temps **faire**
    Choisir a de manière aléatoire
    Effectuer l'action a ; observer r, s'
    $Q(s,a) \leftarrow (1-\alpha_t) * Q(s,a) + \alpha_t * [r + \gamma \max_{a' \in \mathcal{A}} Q(s',a')]$
    $s \leftarrow s'$
fin pour

- La **stratégie d'exploration**, c'est-à-dire le choix aléatoire d'action servant à l'acquisition de nouvelles connaissances. Le plus simple est de lui faire suivre une simple probabilité de distribution uniforme, mais on peut *orienter* l'exploration pour la rendre plus efficace : choisir des actions qui vont mieux renseigner que d'autres sur les bonnes décisions à prendre. L'exploration peut par exemple être simplement améliorée en se basant sur la comptabilisation des paires état-action déjà rencontrées [Thrun, 1992], sur les variations de la fonction Q [Moore, 1990], ou sur une mémoire des essais les plus récents [Sutton, 1990].
- Le **dilemme exploration/exploitation**, c'est-à-dire le compromis qu'il faut régler entre profiter immédiatement des connaissances acquises et les approfondir pour améliorer le gain à plus long terme. Diverses méthodes existent pour gérer ce dilemme (voir sur le sujet [Fabiani *et al.*, 2001]. Nous nous limiterons à une méthode souvent rencontrée et dont nous nous servirons : celle du Q-learning boltzmannien ([Barto *et al.*, 1991; Watkins, 1989]...). Cette méthode consiste à choisir l'action à effectuer selon une distribution de Boltzmann :

$$P(a) \quad = \quad \frac{e^{\frac{Q(s,a)}{T}}}{\sum_{b \in \mathcal{A}} e^{\frac{Q(s,b)}{T}}} \tag{2.19}$$

  où T est un paramètre dit de température. Une température élevée permet d'obtenir un choix quasi-équiprobable, alors qu'en faisant tendre la température vers 0, on s'approche d'un choix déterministe. Il faudra donc définir l'évolution de T, souvent une décroissance vers une valeur finale non nulle.

Le Q-learning boltzmannien permet donc de moduler exploration et exploitation, et permet ainsi d'éviter de distinguer une phase de conception de la politique d'une phase d'exécution. De plus, en adoptant une température limite et un α limite non nuls, on perd certes la propriété de convergence qui restait jusqu'ici valable, mais l'agent peut alors s'adapter de manière continuelle, ce qui va être intéressant dans le cas d'un PDM non homogène, c'est-à-dire dont la fonction de transition n'est pas stationnaire.

Ce qu'il est important de noter, c'est que le Q-learning théorique est une base sur laquelle peuvent être élaborées différentes approches telles que celles citées ci-dessous.

2.2.2.4 D'autres algorithmes

Le Q-learning n'est qu'un premier exemple d'algorithme d'apprentissage par renforcement sans modèle, mais il reste une référence et permet un certain nombre de variantes liées à la gestion de l'exploration, comme cela vient d'être dit.

Le Q-learning se classe en fait dans les méthodes d'apprentissage de *différences temporelles* (temporal difference (TD) learning), du fait que la mise à jour de l'évaluation de Q est effectuée à chaque pas de temps. Une autre famille d'approches est celle des méthodes *Monte-Carlo*, qui consistent (dans le cadre des PDMs) à n'estimer Q qu'à travers une longue phase d'expérimentation.

Une présentation complète des ces deux grandes familles de méthodes d'apprentissage sans modèle est faite dans [Sutton *et al.*, 1998]. Nous nous contenterons, pour illustrer le fait que d'autres algorithmes existent dans notre cadre de travail, d'en citer deux, lesquels gardent une proche parenté avec le Q-learning :

- **SARSA** : Ici, une expérience consiste non seulement en s, a, r et s', mais aussi en l'action a' qui va être choisie depuis l'état s' (en suivant la stratégie d'exploration adoptée). La formule de mise à jour de Q de l'algorithme 2 est alors remplacée par :

$$Q(s,a) \leftarrow (1-\alpha_t) * Q(s,a) + \alpha_t * [r + \gamma Q(s',a')] \tag{2.20}$$

  Le principal effet de cette modification est que la politique obtenue tient compte de la stratégie d'exploration suivie et est donc plus sure. En effet, si cette stratégie consiste en un taux ϵ d'actions au hasard, les autres actions suivant la politique déterministe déduite de Q (stratégie ϵ-glouton), cette politique déterministe va avoir tendance à compenser par exemple les situations dangereuses encourues du fait des actions aléatoires (ne pas rester près d'un précipice si on risque d'y tomber).
- **TD**(λ) : Dans ce second cas, l'effet d'une récompense r n'est pas restreint à la paire état-action qui vient de passer, mais est transmise (avec un taux de diffusion $\lambda \in]0,1[$). Cela permet d'accélérer l'apprentissage par rapport au Q-learning, lequel ne permettait cette diffusion qu'à travers un nombre bien plus grand de passages dans chaque paire état-action. Il s'agit alors d'une approche intermédiaire entre les méthodes de différences temporelles et les méthodes Monte-Carlo.

2.2.3 Limitations des PDM

Différents aspects peuvent être perçus comme des limitations des processus de décision markoviens finis. Nous présentons ici ceux qui nous paraissent les plus notables, en mettant en avant ceux qui vont s'avérer les plus importants pour nous.

Finitude

Le premier problème est lié au fait de travailler sur des ensembles d'états et d'actions finis, sachant que des problèmes plus réalistes tels que celui de la voiture sur la colline [Sutton *et al.*, 1998; Munos, 1997; Baxter et Bartlett, 2001; Scherrer, 2003] concernent des contrôles dans des espaces continus. On cherche alors en général à se ramener au cas fini, en discrétisant par exemple l'espace continu de façon à retrouver les informations suffisantes à une bonne prise de décision (car on n'a plus accès qu'à une observation de l'état réel).

Ce problème, malgré son importance, ne sera pas pris en considération par la suite.

Propriété de Markov

Une deuxième difficulté vient de ce que, dans de nombreux cas, les observations dont est capable l'agent ne sont que partielles, et ne permettent donc pas l'accès à une information suffisante sur le système (ce que l'on a appelé jusqu'ici un état). Dans cette situation très courante des processus de décision markoviens partiellement observables (PDMPO), les algorithmes vus jusqu'ici et fonctionnant sur des états ne marchent plus (s'il y avait convergence, ce serait sans garantie d'optimalité, même localement), la propriété de Markov (équation 2.1 dans la section 2.2.1.2) n'étant plus nécessairement vérifiée.

Remarque : La propriété de Markov est en général reliée à la fonction de transition. Toutefois, si la récompense de l'agent n'est pas liée à son observation, on peut aussi rencontrer des problèmes dans lesquels c'est cette récompense qui est non-markovienne [Thiébaux *et al.*, 2002].

La suite de ce chapitre (section 2.3) va principalement s'intéresser à ce problème de la perte de la propriété de Markov, puisque nos agents s'y trouveront toujours confrontés.

Stationnarité

Un troisième point, bien moins souvent sujet à discussion, est celui de l'homogénéïté (la stationnarité) du PDM. En effet, la fonction de transition $\mathcal{T}$ pourrait dépendre de l'instant t où elle est considérée, de manière à modéliser un phénomène saisonnier ou d'usure. Cela n'empêche pas la propriété de Markov d'être vérifiée : les probabilités de transition dépendent certes de l'instant, mais pas des événements passés.

Le problème est qu'on ne peut apprendre un comportement par essais-erreurs si les règles du jeu peuvent encore changer. La thématique abordée dans la sous-section suivante explique l'intérêt que nous portons à cette situation.

[Spécificité des cadres multi-agents]

Sans parler nécessairement de système multi-agents coopératif, supposons que plusieurs agents évoluent dans un même environnement. Supposons de plus que ces agents s'adaptent à travers une méthode d'apprentissage par renforcement. Un agent percevant les autres comme faisant partie de l'environnement, et l'apprentissage faisant évoluer les comportements de ces autres agents, la fonction de transition de l'environnement va être perçue comme non stationnaire, ainsi que l'illustre la figure 2.7.

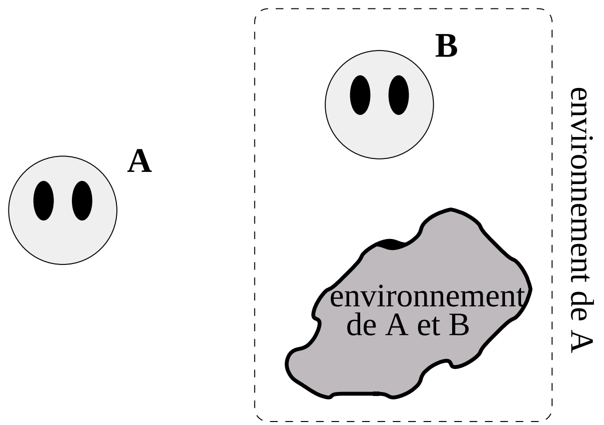


FIG. 2.7 – Ici, l'agent A perçoit B comme faisant partie de l'environnement. Donc si le comportement de B évolue, il en est de même de la fonction de transition $\mathcal{T}$ (vue par A).

On peut en fait avoir deux visions différentes du problème (on suppose ici le cas à deux agents A et B présenté sur la figure 2.7) :

1. *Problème markovien non-stationnaire* : A croît connaître exactement l'état du système (position de B et données relatives à l'environnement commun à A et B), mais la loi d'évolution de l'environnement de A n'apparaît pas comme stable (c'est le point de vue qui a déjà été exprimé).
2. *Problème non-markovien stationnaire* : A ne fait qu'observer l'environnement et n'a pas accès à toutes les informations utiles puisqu'il ne perçoit pas l'état interne de B : sa politique (ou sa table de Q-valeurs si l'on utilise un Q-learning). En effet, la loi d'évolution dictée par le Q-learning est markovienne : la règle de mise à jour ne tient compte que de la situation courante. Il y a donc un processus de décision markovien homogène (stationnaire) sous-jacent derrière ce dont A fait l'expérience, mais A n'a accès qu'à un processus *non*-markovien.

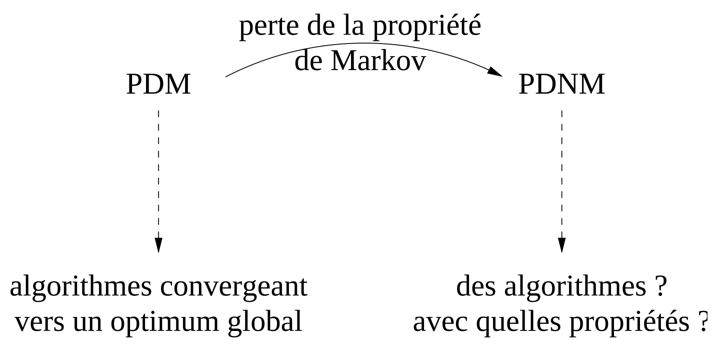
En somme, dans ce problème lié aux apprentissages simultanés d'agents, on peut aussi bien rejeter la faute sur la loi de transition $\mathcal{T}$ qui ne serait pas stationnaire que sur les observations qui ne seraient que partielles.

On retiendra que ce problème de non-stationnarité d'un PDM (qu'il soit lié à un cadre multi-agents ou à toute autre raison) est en général abordé en laissant une part d'adaptation dans le comportement de l'agent (voir à ce propos [Sutton, 1990] par exemple). Là où l'on pourrait se permettre dans un cas stationnaire de faire tendre le taux α de mise à jour de Q vers 0 (pour reprendre l'exemple du Q-learning), il faut ici déterminer un taux limite non nul en adéquation avec la "vitesse" des changements possibles de $\mathcal{T}$. Ce procédé a d'ailleurs déjà été évoqué en section 2.2.2.3.

2.2.4 Bilan

Nous avons vu dans cette section le formalisme des processus de décision markoviens et certains algorithmes (plus particulièrement l'algorithme sans-modèle Q-learning). L'intérêt de ce formalisme vient de ce que la propriété de Markov, vérifiée par tout PDM, donne des garanties de convergence vers des optima globaux pour les algorithmes théoriques présentés.

Malheureusement la propriété de Markov ne sera pas réaliste pour nous, du fait, comme on vient de le voir, du cadre multi-agents, et surtout des perceptions partielles des agents. On va donc regarder si des algorithmes d'apprentissage existent dans des situations non-markoviennes, et plus particulièrement s'il en est qui respectent le cadre que nous avons fixé (principalement l'absence de mémoire à court terme et de modèle de l'environnement). La démarche suivie est illustrée par la figure 2.8.



Pour l'instant, on a examiné le fonctionnement d'algorithmes dans le cas des PDM (pour arriver au Q-learning qui permet de se passer de modèle). Maintenant, c'est vers le cas non-markovien qu'on va se diriger, en cherchant toujours à ne pas utiliser de modèle ou de mémoire à court terme.

FIG. 2.8 – Chemin suivi dans ce chapitre sur l'apprentissage par renforcement.

2.3 Processus de Décision Non-Markoviens

Nous venons de le redire en section 2.2.3 : les agents avec lesquels nous comptons travailler se retrouveront dans une situation non-markovienne, ce qui nous incite à faire usage d'algorithmes de recherche de politique adaptés à de tels *processus de décision non-markoviens* (PDNM).

Pour rappel, le fait que le système ne respecte pas la propriété de Markov (vue en section 2.2.1.2) signifie que la probabilité de transition d'un état s à un autre s', étant donnée l'action a effectuée, dépend de l'historique des états et actions précédents :

$$\exists (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S} \text{ tel que :}$$
$$P(s_{t+1} = s' \mid s_t, a_t) \neq P(s_{t+1} = s' \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_1, a_1, s_0, a_0). \quad (2.21)$$

D'ailleurs, ayant jusqu'ici parlé d'*états* comme de perceptions suffisantes pour prendre une décision optimale sans autre information, nous emploierons le mot *"observation"* dès que la propriété de Markov ne sera pas assurée (la formule 2.21 ci-dessus sera l'exception à cette règle).

La suite du chapitre va tout d'abord préciser ce que l'on entend par processus de décision non-markovien en section 2.3.1, puis présenter dans les sections 2.3.2 et 2.3.3 des méthodes d'apprentissage par renforcement qui peuvent être employées dans ce cadre, en insistant sur celles qui répondent à nos besoins.

2.3.1 PDMPO et autres PDNM

2.3.1.1 Processus de décision markovien caché

Supposons un agent dont le processus de décision, basé sur ses observations o, n'est pas markovien. Si l'on décide de tenir compte de *tout* l'historique des observations et actions passées dans une nouvelle forme d'observations ω (formule 2.22), alors le nouveau processus de décision considéré est, lui, markovien (c'est-à-dire vérifie l'équation 2.1), puisque toute observation ω_t résume les observations $\omega_{t'}$ et actions $a_{t'}$ pour tout $t' < t$:

$$\omega_t = (o_t, a_t, \underbrace{o_{t-1}, a_{t-1}, \ldots, \overbrace{o_1, a_1, \underbrace{o_0, a_0}_{\omega_0}}^{\omega_1}}_{\omega_{t-1}}) \quad (2.22)$$

On peut donc, en théorie, toujours se ramener d'un processus de décision non-markovien à un processus de décision markovien, mais au prix d'un ensemble d'états qui peut être de cardinalité infinie. Nous faisons le choix de nous limiter aux cas où il existe un tel PDM sous-jacent qui soit *fini* (et donc pas fondé sur les historiques).

2.3.1.2 Définition

On parle alors de *processus de décision markovien partiellement observable* (PDMPO) [Smallwood et Sondik, 1973; Monahan, 1982], défini par un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} \rangle$ où :

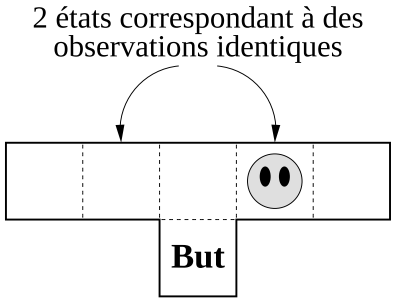
- $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ est un processus de décision markovien (qu'on qualifiera de "sous-jacent"),
- Ω est un ensemble d'*observations*, et
- $\mathcal{O} : \mathcal{S} \times \Omega \to [0; 1]$ est une *fonction d'observation*. Elle donne la probabilité de l'observation o sachant l'état s par :

$$\mathcal{O}(s, o) = P(o_t = o \mid s_t = s) \quad (2.23)$$

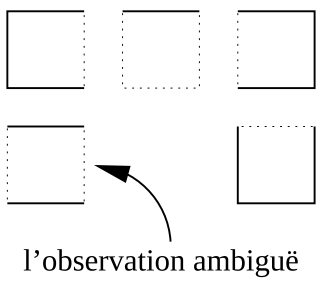
Comme dans le cas des PDM, c'est encore l'espérance de gain moyen qu'on va chercher à maximiser (voir équation 2.2).

Exemple

La figure 2.9 fournit un exemple simple de problème modélisable par un processus de décision partiellement observable. Rappel : l'agent est supposé dénué de mémoire, et ne peut donc agir qu'en fonction de son observation immédiate.



a- Le labyrinthe partiellement observé.



b- Les observations ici possibles.

Ici (image a), on peut reprendre la description du problème de labyrinthe sous la forme d'un PDM vue en section 2.2.1.1, en y adjoignant des observations (image b) n'indiquant à l'agent que la présence de murs dans les quatre directions cardinales (Nord, Sud, Est et Ouest). La fonction d'observation reliant états et observations est intuitive (et ici déterministe).
Dans l'exemple de labyrinthe présenté, deux états se trouvent associés à une même observation. Ainsi, les perceptions immédiates de l'agent ne lui suffisent pas à décider de la bonne action à accomplir.

Fig. 2.9 – Un labyrinthe partiellement observable.

2.3.1.3 Pourquoi d'autres algorithmes ?

Une question qu'il est intéressant de se poser est de savoir dans quelle mesure les algorithmes vus en section 2.2 ne sont plus valables. Si les hypothèses requises dans la démonstration des théorèmes de convergence des algorithmes ne sont certes plus vérifiées, encore faut-il prouver que ces algorithmes peuvent ne pas être optimaux.

Le travail principal que nous recommandons sur ce sujet est celui exposé dans [Singh *et al.*, 1994], lequel montre clairement combien $TD(0)$ et le Q-learning peuvent être inadéquats pour apprendre un comportement basé sur les observations immédiates du PDMPO, lequel comportement doit alors être cherché dans l'espace complet des politiques stochastiques.

Exemple

Pour donner un simple exemple, on peut reprendre celui utilisé par [Singh *et al.*, 1994] pour illustrer le fait que la politique optimale (sur les observations, sans mémoire) peut ne pas être déterministe. Comme le représente la figure 2.10, il s'agit d'un PDMPO à deux états ($1a$ et $1b$) cachés par une unique observation. Les deux politiques déterministes possibles sont d'effectuer soit toujours l'action A, soit toujours l'action B. Mais dans un cas comme dans l'autre, l'agent se trouve rapidement recevoir indéfiniment la récompense négative $-R$.

En l'occurence, la politique optimale est de choisir de manière équiprobable l'une ou l'autre action, ce qui permet d'obtenir une récompense moyenne nulle plutôt qu'une récompense moyenne

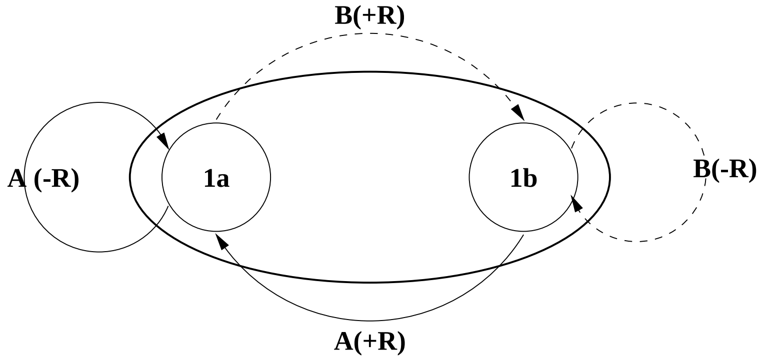


FIG. 2.10 – Un PDMPO pour lequel la politique optimale n'est pas déterministe (tiré de [Singh *et al.*, 1994])

négative ($-R$ dans les pires cas que constituent les politiques déterministes).

Cet exemple justifie clairement le besoin d'outils de recherche de politiques stochastiques, besoin auquel va essayer de répondre la section suivante à travers la présentation de méthodes classiques.

2.3.2 Méthodes classiques avec estimation d'état

La première question qui se pose est de savoir si l'on peut (ou veut) se ramener à un processus de décision markovien (sous-jacent) ou pas. On va ainsi distinguer les algorithmes *avec* et *sans* estimation d'état.

La première approche est donc d'essayer de se ramener à un PDM sous-jacent. Deux situations sont alors possibles (selon les connaissances dont dispose l'agent) :

1. Avec modèle : l'agent connaît le PDM en question $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$, ainsi que la fonction d'observation $\mathcal{O}$, et essaye de trouver dans quel état il se trouve.
2. Sans modèle : l'agent ne connaît pas de tel PDM, mais essaye d'en construire un.

Note : On pourrait aussi supposer un modèle incomplètement connu et que l'on cherche à apprendre $\mathcal{T}$ ou $\mathcal{O}$, mais ce ne sera pas le cas ici.

Avec modèle

L'historique des actions et perceptions de l'agent ne sera généralement pas suffisant pour connaître l'état du système dans le PDM sous-jacent connu. Par contre, on peut estimer une probabilité de distribution sur l'ensemble des états possibles, appelée "*état de croyance*" (*belief state*). On peut alors travailler sur un nouveau PDM défini sur l'ensemble des états de croyance.

Le problème est que ces états de croyance forment un ensemble potentiellement infini (continu). Il faut donc résoudre deux difficultés : estimer l'état courant (voir l'exemple simple de la table 2.1), et trouver un algorithme applicable dans l'espace continu des états de croyance. Nous ne développerons pas plus ce domaine, mais conseillons la lecture de [Littman, 1994b] et [Cassandra, 1998].

Sans modèle

Une autre approche est d'essayer de créer un modèle qui soit markovien en se basant sur l'historique. On parle alors d'apprentissage par renforcement *basé sur la mémoire*.

TAB. 2.1 – Exemple d'estimation d'états de croyance

historique	état de croyance
$(o_0 = 1)$	$P(1a) = P(1b) = 0.5$
$(o_1 = 1, a_0 = A, o_0 = 1)$	$P(1a) = 1,\ P(1b) = 0$
$(o_t = 1, a_{t-1} = A, \dots)$	$P(1a) = 1,\ P(1b) = 0$

Dans ce cas très simple, la dernière action permet de déterminer l'état courant. L'observation, toujours 1, n'apporte aucune information. Par contre, une politique déterministe optimale est désormais accessible.

Si le problème considéré est un processus de décision markovien d'ordre relativement peu élevé (c'est-à-dire que la connaissance d'un historique de longueur 2 ou 3 est une information suffisante pour connaître au mieux l'évolution du système), alors on pourra aisément se ramener à un PDM simple. Par contre, avec des ordres plus élevés, ou simplement des ensembles d'observations un peu grands, on se retrouve facilement confronté à une explosion combinatoire de l'espace d'états.

Dans le but de pallier cette difficulté, un certain nombre de travaux s'orientent vers la recherche d'un historique suffisant à une prise de décision optimale. On ne citera que les algorithmes de croissance d'un arbre d'historiques suffisants de [Dutech, 1999] ou [McCallum, 1995] pour exemples.

Note : Les lecteurs intéressés par ces approches que l'on a dites “avec estimation d'état” (on devrait plutôt dire “utilisant une mémoire”), peuvent aussi lire [Littman et R. Sutton, 2001], papier dans lequel sont présentées de nouvelles idées sur le sujet.

2.3.3 Méthodes classiques sans estimation d'état

La seconde approche revient à abandonner l'idée de se ramener à un processus de décision qui soit markovien, et donc à chercher une politique stochastique qui peut n'être qu'un optimum local (comme dans nombre d'algorithmes d'optimisation, seuls des hypothèses particulières permettent de s'assurer de la globalité de l'optimum atteint). On parle alors de recherche directe de politique.

Note : Attention, l'optimum global atteignable *sans* estimation d'état est la plupart du temps moins bon que la meilleure politique *avec* estimation d'état. La notion d'optimalité est relative aux informations auxquelles l'agent a accès.

Les approches *avec* estimation d'état sont assez laborieuses à mettre en œuvre : soit il faut connaître le PDM sous-jacent du problème traité, soit on ne dispose que de méthodes souvent imparfaites. Il est donc plus simple pour nous d'en rester à des algorithmes se passant d'une estimation de l'état caché. Ainsi, les trois sections qui suivent présentent trois algorithmes (sans estimation d'état donc) qui nous ont intéressés au cours de cette thèse.

2.3.3.1 Q-learning boltzmannien adapté

Partant du Q-learning boltzmannien, on peut faire le simple choix de l'utiliser sur les observations accessibles (donc un problème non-markovien), en l'adaptant de manière à ne pas “trop” tendre vers une politique déterministe [Dutech, 1999].

Température

Cette adaptation se fait en déterminant une température limite non nulle T_{lim}. Mais l'algorithme ainsi obtenu ne donne aucune garantie de convergence, serait-ce vers un optimum local. Il pose de plus le problème du choix de T_{lim}, lequel pourrait sans doute s'aborder par un algorithme d'optimisation simple tel qu'un recuit simulé ou une descente de gradient.

Estimation des Q-valeurs

De plus, pour avoir une meilleure évaluation des Q-valeurs, on peut vouloir remplacer la formule de mise à jour des Q-valeurs. En effet, pour une politique stochastique π suivie (sur un espace d'états), le terme de maximisation doit être modifié comme suit, de manière à avoir une estimation correcte :

$$Q(s,a) \quad \leftarrow \quad (1-\alpha_t) * Q(s,a) + \alpha_t * [r + \gamma \sum_{a' \in \mathcal{A}} \pi(s,a') * Q(s',a')] \tag{2.24}$$

Toutefois, cette modification pose deux problèmes. Le premier est qu'il n'est pas assuré que l'on obtienne le bon résultat en travaillant sur des observations. Le second est que cette formule a pour but d'estimer les Q-valeurs associées à une politique donnée, pas d'améliorer cette politique comme c'était le cas auparavant (la maximisation assurait d'aller vers une politique *déterministe* optimale).

Bilan

C'est principalement par facilité de mise au point (simple réutilisation d'un algorithme classique dans le cadre PDM) que l'on emploie ce Q-learning boltzmannien adapté. Mais il peut donner des résultats suffisants sur des applications pour lesquelles on ne cherche pas nécessairement le "meilleur" comportement.

Note : On peut encore améliorer cet algorithme, en utilisant par exemple des traces d'éligibilité, comme proposé dans [Loch et Singh, 1998]. Nous avons préféré nous contenter ici de sa version classique.

2.3.3.2 Montée de gradient de Baxter

Les deux algorithmes qui restent à présenter permettent de faire une recherche dans l'espace complet des politiques stochastiques, et conduisent à des optima locaux. On parle dans ce cas de recherche directe de politique, approche introduite par [Williams, 1987; 1992], et améliorée à travers les travaux de [Baird, 1999; Baird et Moore, 1999]. Nous allons commencer ici par un algorithme montrant bien que le problème posé est un problème d'optimisation classique, puisqu'il se fonde sur le principe de la descente de gradient. Il s'agit de la *montée de gradient* présentée dans [Baxter et Bartlett, 2001; Baxter *et al.*, 2001].

Rappel sur l'opérateur "gradient"

Le gradient[21] d'une fonction continue et dérivable f définie sur K variables $x_1, \ldots, x_K$ est la fonction qui à $x_1, \ldots, x_K$ associe le vecteur :

$$\overrightarrow{\nabla} f(x_1, \ldots, x_K) \quad = \quad \begin{pmatrix} \frac{\partial f}{\partial x_1}(x_1, \ldots, x_K) \\ \vdots \\ \frac{\partial f}{\partial x_K}(x_1, \ldots, x_K) \end{pmatrix} \tag{2.25}$$

Si les variables sont toutes définies sur $\mathbb{R}$ et que la fonction f est à valeurs dans $\mathbb{R}$, le gradient de f en un point $\overrightarrow{x} = (x_1, \ldots, x_K)$ est un vecteur dont une propriété est d'indiquer la direction dans laquelle f croît le plus vite (direction de plus grande pente).

[21] Sur le gradient, voir `http ://mathworld.wolfram.com/Gradient.html et [Arfken, 1985a; 1985c].`

Principe de la descente de gradient

La méthode classique de *descente de gradient*[22] a pour but de trouver un point en lequel une fonction continue et dérivable $f : \mathbb{R}^n \rightarrow \mathbb{R}$ est minimale. L'idée est de parcourir une suite de points $(\overrightarrow{x_t})_{t\in\mathbb{N}}$ se rapprochant du point le plus bas de la surface.

A l'étape t, on part du point $\overrightarrow{x_t} \in \mathbb{R}^K$, et on suit le vecteur donné par le gradient de f en $\overrightarrow{x_t}$: $\overrightarrow{\nabla} f(\overrightarrow{x_t})$, lequel indique la direction de plus grande pente de f. Ainsi on passe à $\overrightarrow{x_{t+1}}$ par :

$$\overrightarrow{x_{t+1}} \quad = \quad \overrightarrow{x_t} - \lambda_t * \overrightarrow{\nabla} f(\overrightarrow{x_t}), \tag{2.26}$$

où $(\lambda_t)_{t\in\mathbb{N}}$ est une suite réelle positive décroissante définie de manière à progresser rapidement aux premières étapes de l'algorithme (en sortant si possible de zones d'optima locaux), puis à affiner le résultat par des pas de plus en plus petits.

La figure 2.11 illustre l'évolution d'une telle suite de points $(x_t)_{t\in\mathbb{N}}$ pour une suite $(\lambda_t)_{t\in\mathbb{N}}$ à valeurs trop petites pour ne pas tomber dans un minimum local. Grâce à une suite $(\lambda'_t)_{t\in\mathbb{N}}$ à valeurs un peu plus grandes, une deuxième suite $(x'_t)_{t\in\mathbb{N}}$ de même point de départ ($x_0 = x'_0$) trouve, elle, le minimum global.

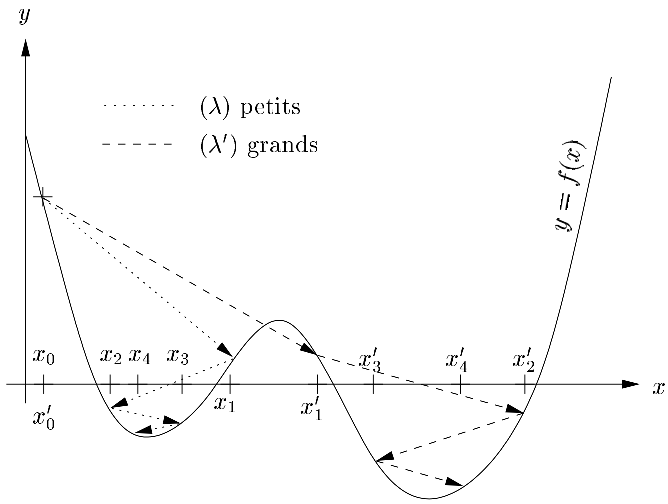


FIG. 2.11 – Comparaison de deux descentes de gradient.

Il ne s'agit que d'une heuristique, puisqu'on risque de tomber dans un extremum local, et que la vitesse de convergence va dépendre du choix de la suite λ pour le problème courant. Si des variantes améliorant cette méthode existent, cette rapide description suffira pour introduire son adaptation à la recherche d'une politique.

Adaptation de notre problème à la montée de gradient

Notes préliminaires :

1. L'objectif qui est poursuivi étant de réaliser un algorithme d'apprentissage par renforcement, c'est désormais un problème de maximisation qui nous intéresse. Nous parlerons donc dès à présent de *montée* de gradient, et non plus de descente.

[22]Sur la descente de gradient, voir `http ://mathworld.wolfram.com/MethodofSteepestDescent.html` et `[Arfken, 1985b]`.

2. De plus, la présentation qui suit est simplifiée, l'objectif étant de faire comprendre le principe de l'algorithme.

Un premier point à éclaircir est de savoir quelle fonction doit être optimisée. Le plus simple est de prendre comme mesure de performance à maximiser l'espérance de gain moyen (objectif présenté en section 2.1.1), laquelle dépendra de la politique adoptée, elle-même fonction d'une paramétrisation $\theta = \{\theta_k\} \in \mathbb{R}^K$ (avec $K \in \mathbb{N}$). C'est dans l'espace de ces paramètres que se fera la recherche.

Etant donnée la paramétrisation θ, on notera la probabilité d'effectuer l'action a en cas d'observation o par : $\mu_a(\theta, o) = P_\theta(a \,|\, o)$. Pour le problème de décision posé, un ensemble de paramètres définit ainsi une chaîne de Markov représentant l'évolution sur l'ensemble des états (et non des observations) de matrice de transition $P(\theta) = [p_{i,j}(\theta)]$.

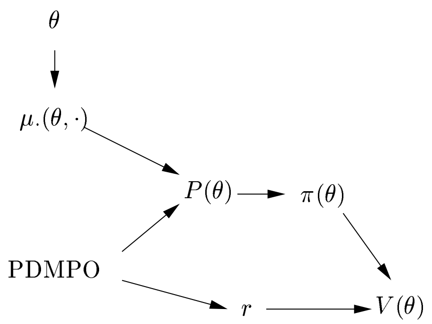
Une hypothèse faite ici est que, pour tout $\theta \in \mathbb{R}^K$, il existe une unique distribution stationnaire notée $\pi(\theta)$ (matrice colonne de probabilités à ne pas confondre avec une quelconque politique), laquelle satisfait l'équation[23] :

$$\pi'(\theta) * P(\theta) \quad = \quad \pi'(\theta) \tag{2.27}$$

Ainsi, si la récompense n'est fonction que de l'état courant[24], en représentant la fonction de récompense par une matrice colonne, le gain moyen une fois l'état stationnaire atteint (gain défini par l'équation 2.2) s'écrit ici :

$$V(\theta) \quad = \quad \pi'(\theta) * r \tag{2.28}$$

La figure 2.12 résume le processus qui vient d'être décrit et qui lie les données du problème à la mesure de performance d'une valeur des paramètres θ.



- PDMPO problème posé
- θ ensemble des paramètres
- $\mu.(\theta, \cdot)$ politique
- $P(\theta)$ matrice de transition de la chaîne de Markov résultant de la politique obtenue
- $\pi(\theta)$ distribution stationnaire sur les états du PDMPO
- r fonction de récompense
- $V(\theta)$ gain moyen une fois en état stationnaire

FIG. 2.12 – Schéma reliant le PDMPO de départ et θ à $V(\theta)$.

Evaluation du gradient

Dans la situation où nous nous trouvons (sans connaissance d'un modèle du monde), on ne peut calculer le gradient de $V(\theta)$ de manière formelle. Seule la politique μ, qu'il nous incombe de définir en fonction de θ, peut être manipulée formellement.

Hypothèses : Dans la suite de cet exposé, voici les hypothèses supposées vérifiées :

1. Pour tout θ, il existe une unique distribution stationnaire $\pi(\theta)$ à long terme (hypothèse déjà énoncée auparavant).

[23] π' indique la transposée de π.

[24] Ces résultats s'étendent à la forme plus générale de récompense d'un PDMPO.

2. Les dérivées
$$\frac{\partial \mu_a(\theta, o)}{\partial \theta_k}$$
existent et les rapports
$$\frac{\left|\frac{\partial \mu_a(\theta,o)}{\partial \theta_k}\right|}{\mu_a(\theta, o)}$$
sont uniformément bornés par $B < \infty$ pour tout $a \in \mathcal{A}$, $o \in \mathcal{O}$, $\theta \in \mathbb{R}^K$ et $k = 1, \dots, K$.
3. La valeur absolue de la fonction de récompense est bornée uniformément par $R < \infty$.

Connaissant μ, l'évaluation du gradient (que l'on notera simplement $\overrightarrow{\nabla}$ ici) peut être faite par itération des deux calculs suivants au fur et à mesure des expérimentations (voir algorithme GPOMDP dans [Baxter *et al.*, 2001]) :

$$\overrightarrow{z_{t+1}} = \beta \overrightarrow{z_t} + \frac{\overrightarrow{\nabla} \mu_{a_t}(\theta, o_t)}{\mu_{a_t}(\theta, o_t)} \tag{2.29}$$
$$\overrightarrow{\Delta_{t+1}} = \overrightarrow{\Delta_t} + r_{t+1} \overrightarrow{z_{t+1}}, \tag{2.30}$$

où o_t est l'observation à l'instant t, a_t l'action choisie, r_{t+1} la récompense reçue et $\beta \in [0, 1)$.

Essayons d'interpréter ces deux formules :

- $\mu_{a_t}(\theta, o_t)$ est la probabilité de choisir l'action a_t pour l'observation o_t et les paramètres θ. $\frac{\overrightarrow{\nabla}\mu_{a_t}(\theta,o_t)}{\mu_{a_t}(\theta,o_t)}$ est donc un vecteur indiquant dans quelle direction (dans l'espace $\mathbb{R}^K$) cette probabilité évolue le plus vite.
- Ainsi, d'après la formule 2.29, $\overrightarrow{z_{t+1}}$ mémorise la direction de modification à laquelle les probabilités des derniers choix effectués sont les plus sensibles. En d'autres termes, la direction de modification de θ qui aura le plus d'effet sur les probabilités liées aux décisions récentes.
- Le terme $\beta \overrightarrow{z_t}$ permet de contrôler un phénomène de persistence plus ou moins important des événements passés dans cette mémoire.
- $r_{t+1}\overrightarrow{z_{t+1}}$ indique dans quel sens il faut aller, d'après la dernière expérience : $\overrightarrow{z_{t+1}}$ est-il un vecteur à suivre pour progresser ou régresser, et dans quelle mesure ?
- En accumulant, comme le fait la formule 2.30, les indications successives des vecteurs à suivre pour améliorer la politique (d'après les expérimentations vécues), on obtient finalement l'évaluation du gradient de V en θ : $\overrightarrow{\Delta_{t+1}}$ (nombre à diviser par $t+1$).

Algorithme utilisé : version en ligne

Sachant désormais comment évaluer le gradient et comment s'en servir pour faire une montée de gradient, on a tous les éléments nécessaires à la mise en application.

L'algorithme choisi n'évalue pas explicitement le gradient de V, mais corrige directement l'ensemble de paramètres θ, et ce, au cours de la vie de l'agent (en ligne). Il s'agit de OLPOMDP, aussi présenté dans [Baxter *et al.*, 2001] et reproduit dans l'algorithme 3.

Paramétrisation choisie

Un point qui doit encore être précisé pour mettre en pratique cette montée de gradient est la forme de paramétrisation choisie. Les auteurs eux-mêmes en proposent une, que nous avons reprise de manière à avoir le maximum de degrés de liberté utiles (un paramètre par

Algorithme 3 OLPOMDP$(\beta, T, \theta_0) \rightarrow \mathbb{R}^K$

Entrées:

- $\beta \in [0, 1)$.
- $T > 0$.
- Des valeurs initiales du paramètre $\theta_0 \in \mathbb{R}^K$.
- Des politiques paramétrées randomisées $\{\mu(\theta, .) : \theta \in \mathbb{R}^K\}$.
- Un POMDP.
- Des tailles de pas $\alpha_t, t = 0, 1, \ldots$ satisfaisant $\sum \alpha_t = \infty$ et $\sum \alpha_t^2 < \infty$.
- Un état de départ arbitraire (inconnu) s_0.

Soit $z_0 = 0$ ($z_0 \in \mathbb{R}^K$).
pour $t = 0$ à $T - 1$ **faire**
    Observer o_t (généré d'après $v(s_t)$).
    Générer une commande a_t d'après $\mu(\theta, o_t)$
    Observer $r(s_{t+1})$ (où le prochain état s_{t+1} est généré d'après $T(s_t, a_t, s_{t+1})$)
    $z_{t+1} \leftarrow \beta z_t + \frac{\nabla \mu_{a_t}(\theta, o_t)}{\mu_{a_t}(\theta, o_t)}$
    $\theta_{t+1} \leftarrow \theta_t + \alpha_t r(s_{t+1}) z_{t+1}$
fin pour

Sorties: θ_T

paire observation-action, ce qui est même plus que nécessaire, mais permet de simplifier les formules). Cette paramétrisation ressemble d'ailleurs aux formules rencontrées dans le Q-learning botzmannien, puisqu'on définit :

$$\theta = \{\theta_{a,o} \in \mathbb{R},\ \forall (a, o) \in \mathcal{A} \times \mathcal{O}\} \tag{2.31}$$

et

$$\forall (a, o) \in \mathcal{A} \times \mathcal{O},\ \mu_a(\theta, o) = \frac{e^{\theta_{a,o}}}{\sum_{a' \in \mathcal{A}} e^{\theta_{a',o}}} \tag{2.32}$$

En remarquant, pour la formule 2.29, la propriété :

$$\frac{\vec{\nabla} \mu_a(\theta, o)}{\mu_a(\theta, o)} = \vec{\nabla} \ln(\mu_a(\theta, o)), \tag{2.33}$$

on peut développer le calcul d'une des composantes de ce vecteur 2.33 :

$$\begin{aligned} \frac{\partial \ln(\mu_a(\theta, o))}{\partial \theta_{a',o'}} &= \frac{\partial \ln \left(\frac{e^{\theta_{a,o}}}{\sum_{b \in \mathcal{A}} e^{\theta_{b,o}}} \right)}{\partial \theta_{a',o'}} \\ &= \delta_{o,o'} * (\delta_{a,a'} - \mu_{a'}(\theta, o')), \end{aligned}$$

où la fonction δ est définie par :

$$\forall (a, b) \in \mathbb{N}^2,\ \delta_{a,b} = \begin{cases} 1 \text{ si } a = b \\ 0 \text{ sinon} \end{cases}.$$

On se retrouve donc bien avec des calculs réalisables et assez légers, si ce n'est qu'il faut les effectuer pour chaque paire observation-action.

Bilan

Si le travail qui mène à cet algorithme est complexe, on obtient un algorithme à peu près aussi simple à mettre en œuvre que le Q-learning boltzmannien mais qui, pour sa part, garantit la convergence vers un optimum local. Une différence notable entre les deux algorithmes est la complexité temporelle, puisqu'ici on effectue à chaque nouvelle expérience des calculs pour tous les paramètres de θ ($|\theta| = |\mathcal{A}| * |\mathcal{O}|$), quand une seule Q-valeur est mise à jour par le Q-learning.

2.3.3.3 Q-learning de Jaakkola & al.

Les deux algorithmes qui viennent d'être présentés (Q-learning boltzmannien adapté et montée de gradient de Baxter) ont tous deux été utilisés pendant la thèse, le second supplantant bien logiquement le premier. Nous allons ici présenter un autre algorithme d'apprentissage par renforcement adapté à un cadre non-markovien (et sans estimation d'état), mais lequel n'a pas été mis en application.

Il aurait pu remplacer la montée de gradient, mais est présenté ici pour la simple raison que nos travaux reviendront sur certaines de ses spécificités ultérieurement.

Idée

Après avoir étudié les difficultés propres aux PDMPO sans estimation d'état, à travers la lecture de [Singh *et al.*, 1994] par exemple, on peut légitimement avoir l'impression que des approches basées sur l'optimisation locale d'une politique (en chaque observation) sont vouées à l'échec.

Pourtant, peu après [Jaakkola *et al.*, 1994b], les mêmes auteurs présentent un algorithme qui, en définissant une nouvelle mesure de performance et surtout de nouvelles grandeurs V et Q, réussit à trouver une politique stochastique optimale (localement) à travers une forme de Q-learning (ce qui explique que nous en parlions comme du "Q-learning de Jaakkola & al.").

Mesure de performance choisie

La mesure de performance choisie est la même que pour la montée de gradient de Baxter : on cherche à maximiser le gain moyen dans le temps, lequel sera noté R^π (ce qui suppose que tout état reste toujours accessible dans la chaîne de Markov). Elle est simplement écrite sous une autre forme :

$$R^\pi = \lim_{N \to \infty} \frac{1}{N} \sum_{t=1}^{N} E(R_t) \tag{2.34}$$

Comme on l'a dit, on revient ici au principe d'évaluer une paire observation-action pour améliorer la politique courante. Dans un cas markovien, l'évaluation s'écrit (pour un état s) :

$$V^\pi(s) = \lim_{N \to \infty} \sum_{t=1}^{N} E(R_t - R^\pi \mid s_t = s) \tag{2.35}$$

Quand t grandit, $E(R_t - R^\pi | s_t = s)$ tend vers 0, puisque l'on se rapproche de l'état stationnaire. Ainsi, les termes les plus importants dans la somme qui permet de calculer $V(s)$ sont les premiers. $V(s)$ représente donc le gain à court terme par rapport au gain moyen R^π (en suivant la politique courante π).

De là, on peut en déduire une définition similaire pour l'évaluation d'une observation, et surtout relier l'évaluation des états du PDMPO à l'évaluation de ses observations comme suit :

$$V^\pi(o) \;=\; \sum_{s\in\mathcal{S}} P^\pi(s \mid o) * V^\pi(s), \tag{2.36}$$

où $P(s\,|\,o)$ est la probabilité d'être dans l'état s si o est observé[25].

Ces définitions s'étendent sans difficultés aux Q-valeurs, puisque l'évaluation d'un couple état(observation)-action ne change le calcul qu'en précisant la première action effectuée, les autres suivant la politique π.

Estimation

Nous n'entrerons pas ici dans les détails du moyen de faire l'estimation des Q-valeurs. Il s'agit d'une méthode de Monte-Carlo, c'est-à-dire d'évaluation statistique par l'expérimentation. L'algorithme proposé dans [Jaakkola *et al.*, 1994b] est même spécifiquement adapté pour améliorer l'estimation continuellement au fur et à mesure de l'expérimentation.

Théorème d'amélioration d'une politique

Le théorème 1 qui suit est la clef de l'algorithme de recherche directe de politique proposé (la preuve ne sera pas reprise ici).

Théorème 1 (Amélioration d'une politique)
Soit la politique stochastique $\pi(a|o)$ qui mène aux Q-valeurs $Q^\pi(o,a)$. Pour toute politique $\pi_1(a|o)$ définissons :

$$J^{\pi_1}(o) \;=\; \sum_a \pi_1(a|o)[Q^\pi(o,a) - V^\pi(o)] \tag{2.37}$$

Le changement de la récompense moyenne qui résulte du changement de la politique courante selon $\pi(a|o) \to (1-\epsilon)\pi(a|o) + \epsilon\pi_1(a|o)$ est donné par :

$$\Delta R^\pi \;=\; \epsilon \sum_o P^\pi(o) * J^{\pi_1}(o) + O(\epsilon^2) \tag{2.38}$$

où les $P^\pi(o)$ sont les probabilités de présence pour les observations associées à la politique courante.

De ce théorème, on peut déduire le schéma algorithmique suivant :

1. Choisir la politique $\pi_1(a|o)$ telle que :
$$J^{\pi_1}(o) \;=\; \max_a [Q^\pi(o,a) - V^\pi(o)]. \tag{2.39}$$
2. Si $J^{\pi_1}(o) > 0$ pour certaines observations o, alors on peut (en ces points) faire évoluer la politique π vers π_1.

[25] Cette probabilité $P(s\,|\,o)$dépend de la politique, contrairement à $P(o\,|\,s)$.

Pour expliquer ce principe de manière plus visuelle, l'idée est que π_1 ne donne pas une meilleure politique que π, mais une direction améliorante. Si, pour une observation o, il existe une action a telle que $V^\pi(o) < Q^\pi(o, a)$, c'est que la politique π peut être améliorée en favorisant le choix de a.

La figure 2.13 illustre ce principe dans une situation où trois actions sont possibles. Dans ce cas, une politique (pour une observation) est représentée par un point sur le triangle défini par les vecteurs unitaires d'un espace à trois dimensions (voir image de gauche). L'image de droite schématise les valeurs $Q^\pi(o, a)$ par des colonnes. La politique π se déplace en direction de π_1, ici vers la colonne la plus haute.

FIG. 2.13 – Le principe d'amélioration de politique suivi

L'algorithme tend vers une position d'équilibre pour laquelle les différentes actions a prises en compte pour l'observation o (c'est-à-dire telles que $\pi(o, a) \neq 0$) ont mêmes Q-valeurs (sinon, il y aurait encore une possibilité d'optimisation).

Bilan

Comme écrit en introduction de cette section 2.3.3.3, cet algorithme est étonnant du fait de sa ressemblance avec ceux utilisés dans le cadre markovien. Mais on notera que le sens des Q-valeurs employées n'est plus le même. Un simple exemple est que, si, pour une observation donnée, deux actions sont nécessaires, alors les Q-valeurs liées à ces deux actions sont égales (en supposant une politique optimale atteinte), même si l'une est prépondérante sur l'autre.

Note : D'autres travaux se sont penchés sur l'utilisation d'une "distance à la récompense moyenne", au moins dans le cas markovien. On pourra par exemple se référer à [Mahadevan, 1996] pour plus de rensignements.

2.3.4 Bilan sur les PDNM

On retiendra ici le fait que la résolution de processus de décision non-markoviens est un problème encore très ouvert, mais que l'on ne peut se satisfaire de supposer l'hypothèse de Markov vérifiée. On a toutefois quelques algorithmes disponibles si les politiques localement optimales (et obtenues sans utilisation de mémoire) restent d'une efficacité acceptable. Dans le cadre de ce mémoire, les algorithmes vus jusqu'ici seront des outils suffisants pour la suite des travaux présentés.

2.4 Conclusion

Dans ce chapitre, nous sommes partis de l'idée d'utiliser le paradigme de l'apprentissage par renforcement pour mettre en œuvre des agents relativement simples, l'A/R étant un moyen de doter des agents d'une capacité d'adaptation (c'est-à-dire d'évolution visant à atteindre un objectif). La figure 2.14 va nous permettre de suivre le chemin parcouru.

Une rapide observation des domaines connexes en section 2.1 (modèles biologiques et outils mathématiques) nous a amené à orienter notre discussion vers les processus de décision markoviens, formalisme qui permet d'écrire des algorithmes dont la convergence vers une solution globalement optimale est prouvée.

Ce champ des PDM a été couvert dans la section 2.2, ce qui a donné l'occasion de préciser quelles formes peut prendre un problème d'apprentissage par renforcement (avec ou sans modèle par exemple). Cela a aussi donné l'occasion de voir quelles sont les limitations de ce formalisme, l'hypothèse d'un système vérifiant la propriété de Markov étant rarement réaliste (propriété qui dit, en résumé, que l'observation courante suffit pour prendre une décision optimale).

On en est donc logiquement venu aux processus de décision non-markoviens et aux algorithmes d'apprentissage associés. Parmi ceux-là, nous avons présenté :

- d'abord rapidement ceux qui reposent sur l'estimation de l'état du système,
- puis plus en détail ceux qui correspondent au type d'agents que nous nous sommes fixés : sans mémoire à court terme ni modèle de l'environnement, avec des perceptions partielles et sans moyens de communication.

Ce parcours a permis de voir quelles formes peut prendre le problème de l'apprentissage par renforcement, quelles en sont les difficultés, et quels outils nous permettrons de travailler dans le cadre que nous nous sommes fixés.

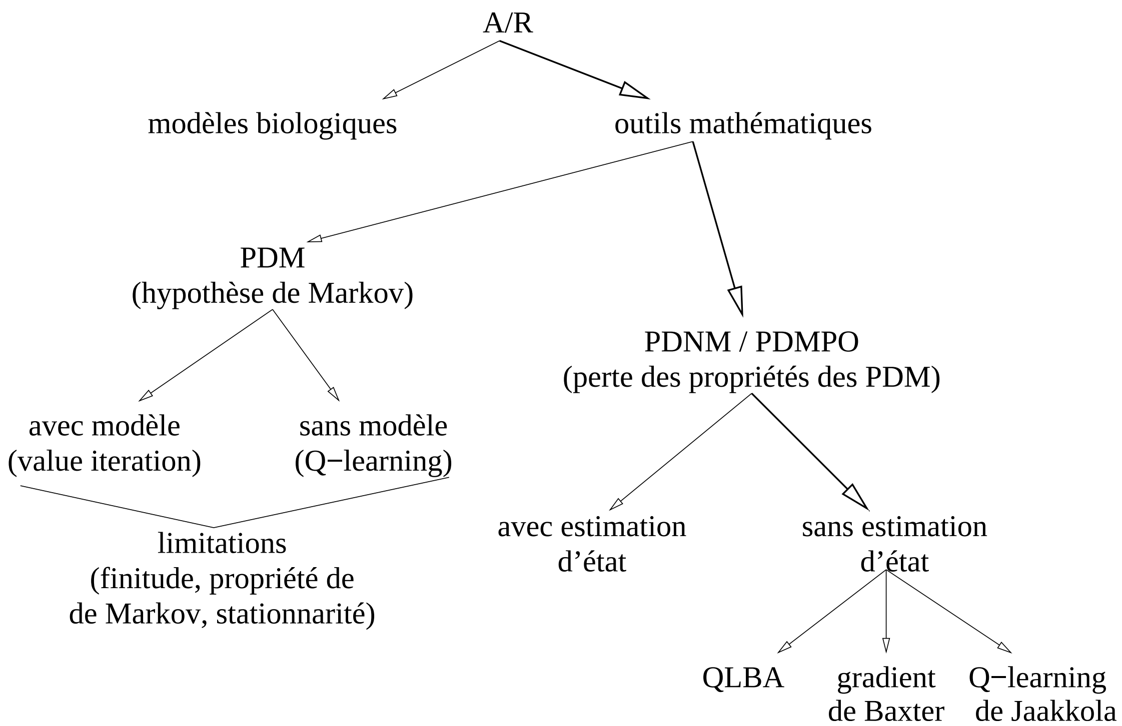


Fig. 2.14 – Parcours suivi dans ce chapitre sur l'apprentissage par renforcement.

Conclusion

Ces deux premiers chapitres ont été l'occasion d'introduire les outils (SMA et A/R) dont il sera fait usage dans la suite de ce mémoire. Au fur et à mesure de cette présentation nous avons précisé à quels aspects nous souhaitons nous attacher plus particulièrement, et mis en avant les problèmes posés par l'une et l'autre approche.

La notion d'agent nous servant de paradigme pour parler d'un système intelligent, la problématique de l'I.A. se ramène pour nous au problème de la conception d'agents. Partant de ce point, on a discuté dans le chapitre 1 :

- de ce que nous appellerons agent (section 1.1), vu comme un système capable de s'adapter, en insistant sur son autonomie, sur les problèmes liés aux incertitudes qui l'entourent et en mettant en évidence le fait que cette notion peut aisément revêtir un caractère social ;
- d'un domaine assez vaste de l'I.A. auquel notre travail va être lié : celui des systèmes multi-agents (section 1.2), domaine qui propose de résoudre des problèmes par des approches collectives ; et
- de ce en quoi consistera pour nous la conception d'agents ou de systèmes multi-agents (section 1.3), l'idée principale étant de réaliser des architectures internes d'agents capables de s'adapter, mais avec des difficultés particulières dans un cas multi-agents (faire que les agents collaborent).

Or la conception d'agents se lie très naturellement à l'idée d'apprentissage par renforcement, puisqu'il s'agit d'adapter un comportement de manière à maximiser un gain, lequel gain peut représenter une grande variété d'objectifs. Le chapitre 2 consacré aux processus de décision markoviens :

- justifie d'abord (section 2.1) le choix de ce formalisme : manipulabilité théorique et gestion de l'incertitude ;
- puis (section 2.2) montre plus en détails les résultats théoriques et outils algorithmiques dont il permet de disposer, mais aussi ses limitations, puisque l'hypothèse de Markov requise est difficile à assurer (du fait de perceptions seulement partielles) ; et
- finalement (section 2.3) présente le domaine des processus de décision non-markoviens qui s'avèrent souvent plus réalistes, mais sont aussi plus difficiles à aborder et ne fournissent que des algorithmes au mieux localement optimaux dont nous nous contenterons.

Ayant présenté de façon assez indépendante SMA et A/R, nos travaux vont désormais pouvoir diversement s'intéresser à l'usage de l'apprentissage par renforcement dans le cadre d'agents coopérants, comme à la réalisation (toujours par A/R) d'agents sous une forme modulaire dont on verra la parenté avec les SMA. Ces deux parties distinctes seront l'occasion de voir comment A/R et SMA peuvent servir l'un à l'autre et réciproquement.

Deuxième partie

Apprentissage progressif dans un cadre multi-agents

Introduction

> « L'important de la pédagogie n'est pas d'apporter des révélations, mais de mettre sur la voie. »
>
> *Un même mystère, Pierre Dehaye*

Dans cette deuxième partie, nous allons nous intéresser à la conception d'agents qui coopèrent dans une tâche commune. De manière générale, comme cela a été dit en section 1.3.2, la réalisation des comportements d'agents indépendants (ils sont indépendants même si leur l'action doit être mesurée au niveau du groupe) est un problème très ouvert. Aucune théorie ne permet aujourd'hui de faire cette mise au point en partant de l'objectif commun et en suivant un processus systématique.

Si les techniques de l'apprentissage artificiel (*Machine Learning*) en général peuvent s'avérer utiles à la conception de systèmes multi-agents [Stone et Veloso, 2000], le cadre de l'apprentissage par renforcement va pouvoir jouer un rôle clef puisqu'il propose des méthodes de conception guidées par un but [Weiß et Sen, 1996].

Le chapitre 3 sera l'occasion de faire un survol des travaux orientés autour de l'emploi du formalisme des processus de décision markoviens pour la conception de SMA, et de voir les possibilités offertes par cette approche, ainsi que ses limitations.

De là, nous proposerons au chapitre 4 une méthode d'aide à l'apprentissage qui se base sur une progression régulière de la complexité de la tâche d'une part, et du nombre d'agents engagés d'autre part. Sans résoudre tous les problèmes constatés au chapitre 3, cette méthode permet d'améliorer les résultats obtenus. Elle sera validée et analysée par des expérimentations présentées en fin de chapitre.

Chapitre 3

PDM pour SMA

Sommaire

Ce chapitre (issu d'un travail commun avec Iadine Chadès) est consacré à un aperçu d'un certain nombre de travaux qui se basent sur l'utilisation de formalismes dérivés des processus de décision markoviens dans un cadre multi-agents. Ces travaux permettront de mieux appréhender les difficultés liées à cette problématique.

Afin de mieux cerner les apports que constituent pour nous ces différents travaux, rappelons que notre souhait est de concevoir des systèmes multi-agents qui soient *coopératifs*, et ce, dans un cadre non-markovien, les différents agents ne disposant que de perceptions partielles, sans moyen de communiquer ni mémoire à court terme.

Du fait de la disparité des approches, l'ordre de présentation adopté est principalement orienté par la chronologie des apparitions respectives des travaux présentés. Mais cet ordre s'avère progresser assez bien vers la problématique qui nous préoccupe. Les thèmes qui vont être ainsi abordés dans le présent chapitre sont (dans les différentes sections qui le composent) :

- (3.1) **Jeux de Markov** : Ce formalisme permet de prendre conscience des phénomènes complexes qui apparaissent dans des problèmes de concurrence tels qu'en économie, et de voir comment l'apprentissage peut y jouer un rôle (le tout ressemblant beaucoup à ce que nous voulons faire).
- (3.2) **MMDP** : Ce second formalisme, tout en restant dans une situation d'observation totale contraire à nos hypothèses, présentera une situation comparable à celle des jeux de Markov, mais ici uniquement dans des cas de coopération.
- (3.3) **Dec-POMDP** : Eux donneront simplement l'occasion d'étudier la complexité du problème de conception d'agents coopérant, mais cette fois-ci avec perceptions partielles, ce qui correspond à notre objectif.
- (3.4) **Descentes de gradient** : On verra un moyen simple et efficace de concevoir de manière centralisée un SMA réactif.
- (3.5) **COIN** : Le problème (jusqu'ici ignoré) du passage d'une récompense de groupe à des récompenses individuelles sera alors abordé.
- (3.6) **Empathie** : Finalement, on présentera une approche de conception de SMA plus pratique que celles vues précédemment (basée sur le principe de modélisation d'autrui).

Note : On adoptera ici les acronymes anglais d'origine.

3.1 Théorie des jeux et jeux de Markov

3.1.1 Théorie des jeux

L'intérêt de certains mathématiciens pour les jeux a conduit au développement de notions telles que probabilités et dénombrement dès le $XVII^{\text{ème}}$ siècle, avec les travaux de Blaise Pascal, Pierre de Fermat ou des frères Jacques et Jean Bernoulli. En plus de ce premier domaine d'étude mêlant hasard et compétition qu'est le jeu, un second est apparu : l'économie (les problèmes posés étant très ressemblant d'un domaine à l'autre en fin de compte). C'est de ces divers développements qu'est apparue la *théorie des jeux* [Neumann et Morgenstern, 1967; Owen, 1982].

3.1.2 Jeux de Markov

Dans le vaste champ de la théorie des jeux, c'est plus particulièrement sur les *jeux de Markov* [Shapley, 1953] (aussi appelés *jeux stochastiques*) que nous allons nous attarder.

Un jeu de Markov considère différents joueurs (qui seront pour nous des agents) qui peuvent être en concurrence, et ne sont en tout cas pas nécessairement en situation de coopération, au sens qu'ils ne suivent pas un objectif commun. Si l'idéal en résolution collective de problèmes est que les agents aient réellement un tel but commun, c'est rarement le cas, tant il est difficile d'exprimer une complète coopération à travers la fonction de récompense.

Nous reviendrons sur ce problème "d'assignement des crédits"(*credit assignement problem*) en section 3.5. Pour l'instant, le cadre des jeux de Markov va nous permettre de mieux comprendre les effets de la concurrence entre agents.

Définition 1 (jeu de Markov)
Un jeu de Markov est défini par un tuple $G = \langle \mathcal{S}, \mathcal{N}, \mathcal{A}_1, \dots, \mathcal{A}_n, \mathcal{T}, u_1, \dots, u_n \rangle$*, où :*
- $\mathcal{S}$ *est un ensemble fini d'états,*
- $\mathcal{N}$ *est un ensemble fini de* n *joueurs,*
- $\mathcal{A}_i$ *est un ensemble fini d'actions possibles pour le joueur* i*. On pose* $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n$ *l'ensemble des actions jointes.*
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *est la fonction de transition du système, laquelle donne la probabilité d'aller d'un état* s *à un état* s' *quand l'action jointe* a *est accomplie.*
- $u_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *est la fonction d'utilité réelle pour le joueur* i *(*i.e. *sa fonction de récompense).*

Ainsi, chaque joueur i tente de maximiser son espérance de gain individuellement, en utilisant le critère de performance γ-pondéré suivant :

$$E\left(\sum_{j=0}^{\infty} \gamma^j u_{i,t+j}\right)$$

où $u_{i,t+j}$ est la récompense reçue j pas de temps dans le futur par le joueur i. On pourrait définir un critère non pondéré, mais dans ce cas tous les jeux de Markov n'ont pas de stratégie optimale [Owen, 1982].

Ce critère mesure la performance d'une stratégie, comme définit ci-après :

Définition 2 (Stratégie (pure) - Vecteur de stratégies (pures))
Dans un jeu de Markov, une stratégie pure (on parle aussi de politique) pour un joueur i *est une fonction* $\pi_i : S \to A_i$*. Un ensemble de stratégies pour tous les joueurs* $\pi = \pi_1 \times \dots \times \pi_n$ *est appelé un vecteur de stratégies (ou vecteur de politiques).*

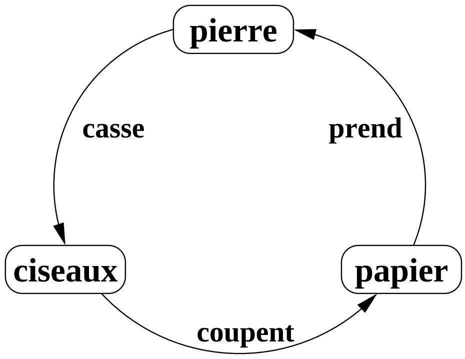
Dans un jeu stochastique, le meilleur comportement à adopter est souvent non-déterministe (se basant sur l'effet de surprise), d'où la définition d'une stratégie *mixte* :

Définition 3 (Stratégie mixte - Vecteur de stratégies mixtes)
Dans un jeu de Markov, une stratégie mixte pour un joueur i *est une fonction* $\pi_i : S \to \Pi(A_i)$ *(qui à un état associe une distribution de probabilité sur les coups possibles). Cette définition s'étend comme précédemment à la notion de vecteur de stratégies mixtes.*

Exemple : Roshambo

Ce premier exemple est ce que l'on appelle un jeu répété : il n'existe qu'un seul état, on n'a donc pas à planifier sur le long terme. Ce jeu, plus généralement connu sous le nom de "pierre-papier-ciseaux", est présenté par la figure 3.1.

Nommons les joueurs A et B. On est ici dans le cas d'un jeu dit *à somme nulle* parce que le gain de l'un est pris comme égal à la perte de l'autre joueur. On peut aborder ce jeu selon différentes situations, dont les trois suivantes (en adoptant le point de vue de A) :



Ce jeu consiste pour chaque joueur à choisir un de ces trois objets, sachant que chacun sera jugé “victorieux” sur l'un des deux autres, comme illustré ci-contre (une flèche représentant la relation “est victorieux sur”).

FIG. 3.1 – Roshambo

1. On sait que B joue plus majoritairement `papier`. Dans ce cas, on a intérêt à jouer systématiquement `ciseaux`, ce qui maximise notre espérance de gain. Dans ce cas où B est supposé avoir une politique fixée, une politique déterministe est optimale.
2. On ne connait pas la manière de jouer habituelle chez B. Le plus sûr est d'opter de manière équiprobable pour chacun des trois coups possibles (si l'on veut une stratégie adoptée de manière définitive).
3. Revenant au premier cas, on peut maintenant supposer que B s'adapte progressivement à notre stratégie. Si nous en faisons de même, les deux joueurs vont en arriver à la stratégie mixte “équilibrée” déjà vue dans le deuxième cas.

Pour les curieux, deux sites sur ce jeu :

`http ://www.cs.ualberta.ca/~darse/rsbpc.html`

`http ://www.worldrps.com/`

3.1.3 Équilibres

Une idée qui apparaît à travers l'exemple du jeu Roshambo est que les joueurs vont se retrouver dans des situations d'équilibre. Nous allons présenter ici deux notions ainsi liées à l'étude des jeux stochastiques.

Equilibre de Nash

L'*équilibre de Nash*, initialement défini pour la théorie des jeux classique, a été étendu aux jeux de Markov. On l'appelle également *équilibre parfait de Markov* et le définit comme suit :

Définition 4 (Équilibre de Nash)
Un vecteur de stratégies π est un équilibre de Nash si et seulement si pour tout joueur i :

$$V_i^{\pi}(s) \quad \geq \quad V_i^{\pi_{-i}\times\pi'_i}(s), \ \forall s \in S, \ \forall \pi'_i$$

(où π_{-i} est le vecteur des stratégies des joueurs autres que i, et les $V_i^{\pi}(s)$ des fonctions d'utilité comme définies dans le cadre de l'apprentissage par renforcement).

En d'autres termes, un vecteur de stratégies est un équilibre de Nash si aucun joueur ne peut améliorer son espérance de gain s'il est le seul à modifier son comportement. Tout jeu de Markov a au moins un équilibre de Nash, dont le calcul est en général très compliqué.

Optimalité de Pareto

Cette notion d'*optimum de Pareto* s'intéresse plus généralement à savoir si le vecteur des stratégies courantes est un "optimum" local ou global.

> **Définition 5 (Optimum de Pareto)**
> *Un vecteur de stratégies π est appelé optimum de Pareto si et seulement si, pour tout autre vecteur π' et tout joueur i, $V_i^{\pi'} > V_i^{\pi}$ implique qu'il existe un agent j pour lequel $V_j^{\pi} > V_j^{\pi'}$ (où les V_k^{π} sont des mesures de performance globales, et non plus par état).*

En d'autre terme, il n'existe pas d'autre vecteur de stratégies qui améliore le gain de la plupart des joueurs sans détériorer le gain de l'un d'entre eux.

Exemple : un jeu de coordination

On s'intéresse ici encore à un jeu n'ayant qu'un seul état, mais en considérant qu'il n'y a qu'un seul coup, et que cette fois-ci A et B sont les actions possibles pour les deux joueurs. On se limite de plus à des stratégies pures.

La figure 3.2 représente dans chaque case les gains de l'un et l'autre joueur. En l'occurence, les joueurs doivent se coordonner pour maximiser leur gain, qui sera toujours le même pour les deux joueurs.

	A	B
A	2,2	0,0
B	0,0	1,1

- Les 2 équilibres de Nash sont entourés d'un trait continu.
- Le seul optimum de Pareto est encadré d'un trait pointillé.

FIG. 3.2 – Un jeu de coordination

On peut ici constater que :
- Si les deux joueurs ont la même stratégie pure, que l'un seulement des deux change sa politique et il est sûr d'avoir un gain moindre (passant de 1 ou 2 à 0). Ainsi, les situations dans lesquelles les joueurs se coordonnent pour adopter la même stratégie sont des équilibres de Nash.
- Aucun vecteur de stratégies n'est meilleur pour les deux joueurs que le cas où tous deux jouent A. Il s'agit donc d'un optimum de Pareto (attention : un optimum de Pareto n'est pas nécessairement un équilibre de Nash, comme le montre l'exemple de la figure 3.3).

3.1.4 Méthodes de résolution pour les jeux de Markov

Nous allons maintenant voir deux travaux qui lient plus précisément l'apprentissage par renforcement avec les jeux de Markov.

Jeux de Markov à somme nulle → apprentissage par renforcement [Littman, 1994a]

Le premier exemple que nous donnons ici est un cas assez particulier, mais dont l'étude permet de bien comprendre les situations de conflits quand deux agents ont des motivations strictement

	nier	trahir
nier	−1,−1	−4,0
trahir	0,−4	−3,−3

Dans ce problème, on a un équilibre de Nash et un optimum de Pareto (notés de la même façon que sur la figure 3.2).

Ici, deux bandits ont été pris, et la police cherche a obtenir leurs aveux. Chacun a le choix entre *avouer* et *nier* le délit. On obtient les situations suivantes :

– Si les deux tiennent leur langue, ils feront chacun un an de prison $\rightarrow (-1,-1)$.
– Si seul l'un des deux trahit, il a une remise de peine, et son complice prend le maximum de quatre ans $\rightarrow (0,-4)$ ou $(-4,0)$.
– Si les deux avouent, c'est trois années que chacun passera derrière les barreaux $\rightarrow (-3,-3)$.

FIG. 3.3 – Le dilemme du prisonnier

concurrentes (jeux à somme nulle tels que Roshambo vu précédemment).

On suppose donc ici que chaque agent connaît non seulement l'état exact du monde, mais en plus sa fonction de récompense, comme celle de l'adversaire. Seul le comportement de son adversaire lui est inconnu.

[Littman, 1994a] présente deux approches pour apprendre par renforcement une stratégie "optimale" : l'une basée sur un modèle de l'autre (hypothèse qu'il joue au mieux), l'autre pas. Voici une explication succinte de leurs principes respectifs :

1. Sans faire l'expérience de la méthode de jeu de l'adversaire, on peut faire l'hypothèse qu'il joue au mieux, ce qui amène à chercher à maximiser notre gain sachant que lui tente de le minimiser. On peut alors modifier les algorithmes de planification tels que *value iteration* (vu en section 2.2.2.2) pour remplacer le terme $\max_{a\in\mathcal{A}}$ par $\max_{\pi\in\Pi(\mathcal{A})}\min_{o\in\mathcal{O}}$ (au lieu de choisir une meilleure politique déterministe, on choisit la meilleure politique stochastique, sachant que l'adversaire cherche à nous nuire). Ainsi, en reprenant les notations de Littman, on passe de :

$$V(s) = \max_{\pi\in\Pi(\mathcal{A})} \sum_{a\in\mathcal{A}} Q(s,a)\pi_a, \text{ où} \tag{3.1}$$

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s')V(s'), \tag{3.2}$$

à :

$$V(s) = \max_{\pi\in\Pi(\mathcal{A})} \min_{o\in O} \sum_{a\in\mathcal{A}} Q(s,a,o)\pi_a, \text{ où} \tag{3.3}$$

$$Q(s,a,o) = R(s,a,o) + \gamma \sum_{s'} T(s,a,o,s')V(s'), \tag{3.4}$$

où o est l'action de l'adversaire.

2. L'autre approche est d'utiliser d'une façon comparable un Q-learning adapté à la présence d'un concurrent. Ce second algorithme a été appelé minimax-Q. La différence avec la première approche est qu'on ne fait pas ici d'hypothèse sur les capacités de jeu de l'adversaire, on s'y adapte.

L'avantage de la deuxième approche est donc de pouvoir s'adapter aux éventuels défauts du joueur adverse (s'il a une préférence pour les ciseaux comme on en a déjà donné l'exemple).

Des difficultés réapparaissent si l'adversaire adopte un comportement similaire : les vitesses d'adaptation de l'un et de l'autre vont entrer en jeu. Mais nous n'irons pas ici plus avant dans l'étude de l'évolution simultanée de deux joueurs "adaptatifs", les situations vues ici montrent déjà bien le rôle que peut jouer l'apprentissage.

Jeux de Markov à somme non nulle → modélisation d'autrui [Hu et Wellman, 1998a][Hu et Wellman, 1998b]

Des travaux tels que ceux présentés dans [Hu et Wellman, 1998a] et [Hu et Wellman, 1998b] sont partis de la même approche "jeux de Markov", mais dans un cas plus orienté coopération d'agents. Il s'agit en effet de considérer n agents (pas nécessairement 2), chacun ayant ses propres gains (indépendants des gains des autres agents, à la différence des jeux à somme nulle)[26]. On adopte donc le point de vue d'un agent qui cherche à maximiser ses gains en considérant que ses congénères font de même.

Le but est d'atteindre un *point fixe* du système, c'est-à-dire une situation d'équilibre dans laquelle aucun agent ne peut améliorer ses résultats sans que les autres changent aussi de comportement (les politiques étant a priori stochastiques). Les agents vont devoir se mettre d'accord sur le point fixe (en l'occurrence un équilibre de Nash) à atteindre, et ce, sans communication.

Un agent a le choix entre deux voies s'il fait de l'apprentissage par renforcement : soit il ne distingue pas les autres agents de l'environnement (on parlera d'agent simplement *compétitif*), soit au contraire il les distingue (leur reconnaît un but), mais cela nécessite de modéliser ces autres agents. Si l'on modélise explicitement un autre agent, on peut même distinguer de manière plus précise des niveaux de modélisation de plus en plus élevés :

- Le niveau 0 correspond à un agent compétitif, c'est-à-dire au simple apprentissage par l'observation du comportement de l'autre (comme s'il faisait partie de l'environnement, en négligeant le fait qu'il a un objectif propre).
- Le niveau 1 au contraire considère que l'autre est compétitif : qu'il a un but, mais qu'il adopte pour sa part une modélisation de niveau 0.
- Un niveau $n \geq 2$ de modélisation est défini par récurrence comme une situation où un agent considère son congénère comme étant de niveau $n-1$.

La figure 3.4 illustre ces différentes situations.

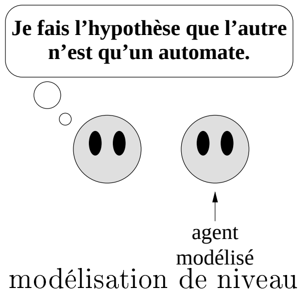
Note : Chaque agent fait une hypothèse sur ses congénères. Très facilement, on va se trouver dans des situations où nécessairement l'un des deux agents se trompe. Prenons l'exemple de deux agents, le premier étant de niveau 2. Soit il se trompe, soit effectivement le second est de niveau 1, auquel cas c'est ce dernier qui se trompe (puisqu'il croit le premier de niveau 0).

Les travaux de Hu et Wellman mettent l'idée de "modélisation de l'autre" en application dans un problème simple de marché boursier, donc non coopératif. La comparaison de différentes situations dans leur système à quatre agents a fait apparaître le fait qu'une modélisation mal faite est bien pire qu'une absence de modélisation. L'idée de méta-raisonnement sous-jacente dans cette approche est intéressante, mais demanderait peut-être une analyse dans un cadre clairement coopératif.

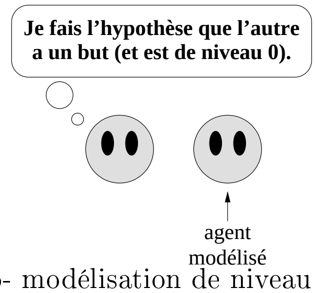
3.1.5 Conclusion

La thématique de la coopération dans un système multi-agents pourrait donner lieu à des travaux approfondis dans le domaine des jeux de Markov, ou même plus généralement de la théorie des jeux. Cette section avait pour simple but de montrer les raisonnements auxquels

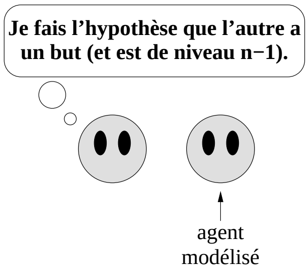
[26] Attention : on se place ici dans un jeu répété, donc un système à un seul état.



a- modélisation de niveau 0



b- modélisation de niveau 1



c- modélisation de niveau n

Dans le cas b, l'agent de droite est supposé avoir un but, mais considère l'agent de gauche comme un automate (ce qui était le point de vue de l'agent de gauche dans le cas a).

FIG. 3.4 – Exemple de trois niveaux différents de modélisation d'autrui.

on peut déjà aboutir en supposant un modèle du monde connu, ainsi éventuellement que les motivations des autres agents.

De manière générale, l'utilisation de jeux de Markov pose problème parce que justement elle requiert au moins une observation complète de l'environnement et éventuellement un modèle. Les cas pratiques ne supportent pas de telles hypothèses, autant pour des raisons de puissance de calcul que de connaissances a priori ou de capacités des capteurs. C'est pourquoi nous avons choisi de faire des hypothèses moins fortes, mais au prix de solutions moins efficaces.

Les sections qui suivent ne concernent pas des domaines aussi vastes, mais des travaux plus ciblés, se rapprochant du cadre SMA et apprentissage par renforcement que nous souhaitons adopter. Nous allons par exemple commencer par des travaux qui, supposant à peu près autant de connaissances que dans les jeux de Markov, ajoutent le fait que les agents sont réellement coopératifs.

3.2 MMDP

[Boutilier, 1996; 1999]

Dans [Boutilier, 1996; 1999] est abordé le problème de la coordination d'agents dans une situation de coopération complète : les agents ont tous la même récompense, et perçoivent tous l'état complet du système.

Nous allons voir le formalisme précis utilisé dans ces travaux, puis les méthodes proposées pour obtenir des ensembles de politiques optimales.

3.2.1 Formalisme

Définition 6 (MMDP)

Un PDM multi-agent M est défini par un tuple $\langle \alpha, \{A_i\}_{i\in\alpha}, S, Pr, R\rangle$ dans lequel :

- *α est un ensemble fini de n agents.*
- *A_i est l'ensemble fini des actions individuelles de l'agent i.*

– *$\mathcal{A}$ est l'espace des actions jointes : $\mathcal{A} = \times A_i$, dont un élément $\langle a_1, \ldots, a_n \rangle$ représente l'exécution simultanée des actions a_i de chaque agent i.*
– *S, Pr, R sont les mêmes que dans un PDM, à l'exception de Pr la fonction de probabilité de transitions qui utilise les actions jointes $\langle a_1, \ldots, a_n \rangle$*

Ce modèle peut être vu comme un PDM avec de grands espaces d'états et d'actions : l'espace des actions devient celui des actions jointes $\mathcal{A}$. On peut aussi le voir comme un jeu de Markov dans lequel tous les agents partagent une même récompense.

3.2.2 Approche de conception centralisée

S'il est possible de concevoir les plans réactifs des agents de manière centralisée avant de passer à la phase d'exécution, et si l'on a accès au modèle de l'environnement (Pr et R) en l'occurence, une approche assez simple est possible.

Il suffit d'appliquer un algorithme classique tel que *value iteration* pour obtenir une politique jointe déterministe $\Pi = \langle \pi_1, \ldots, \pi_n \rangle$. On peut ensuite passer à la phase d'exécution en assignant à chaque agent la partie de la politique qui le concerne (du fait des observations totales, ça ne pose guère de problème). La figure 3.5 illustre cette procédure.

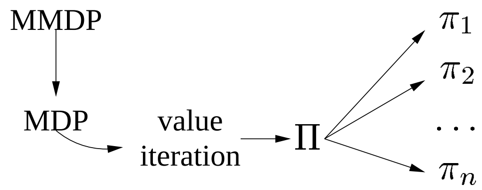


FIG. 3.5 – Conception centralisée des politiques des agents composant un MMDP.

3.2.3 Approche de conception décentralisée

Une situation plus difficile est celle où les agents sont autonomes dès la phase de mise au point de leurs comportements respectifs. Car s'ils peuvent tous adopter un même raisonnement pour trouver la fonction de valeur optimale "jointe" du PDM, et donc les différentes politiques optimales, encore faut-il que tous se mettent d'accord sur la politique optimale à adopter.

Exemple : La figure 3.6 donne un exemple de problème de coordination à deux agents (A_1 et A_2). On a ici un problème à 5 états et 2 actions possibles à chaque instant pour chaque agent (a et b). Pour illustrer le fonctionnement des transitions, $\langle a; * \rangle$ signifie ici que de l'état s_1 le système va dans l'état s_2 si A_1 effectue l'action a (et ce, quelle que soit l'action de A_2). C'est dans l'état s_2 que les agents doivent ici se coordonner, de manière à effectuer tous deux la même action et aller en s_3.

Pour qu'un groupe d'agents se mette d'accord sur une politique jointe à accomplir, trois types de mécanismes de coordination peuvent être adoptés :

– **communication** : Les agents peuvent échanger des messages et se mettre ainsi d'accord sur la politique à suivre. Mais cela suppose d'une part une possibilité de communiquer, et d'autre part la définition d'un protocole de communication, c'est-à-dire d'une partie du comportement des agents.
– **convention** : Les agents connaissent des règles qui permettent de savoir quelle action jointe adopter. Ici, les deux agents peuvent supposer que A_1 choisit la première action, et que A_2 décide en conséquence.

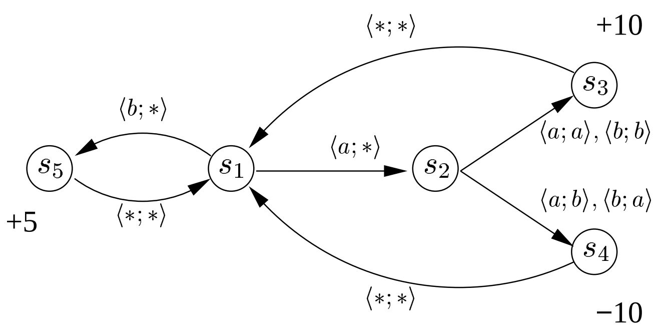


FIG. 3.6 – Exemple d'une situation posant un problème de coordination.

- **apprentissage** : Les agents apprennent par essais-erreurs, ne fixant une action jointe que s'ils tombent par hasard dessus. Il s'agit d'un algorithme randomisé, dont nous allons voir maintenant une version détaillée.

Value iteration étendu

Pour mettre en œuvre cette coordination randomisée, l'algorithme proposé se base sur *value iteration*, en étendant l'espace d'états. En effet, chaque agent va appliquer de son côté l'algorithme de planification (sur le PDM joint), en notant les états dans lesquels il y a plus d'une action *individuelle potentiellement optimale* (action PIO). L'extension se fait en rajoutant à tout état s l'information disant si la coordination a déjà été faite, en prenant la notation $\langle s, c \rangle$, où c indique l'action choisie ou U si la résolution n'a pas encore été faite (U pour *uncoordinated*).

La méthode de recherche d'une coordination par essais-erreurs peut employer diverses approches. On peut ainsi citer l'algorithme *fictitious play* [Shapley, 1964], lequel fait l'hypothèse que les autres agents vont probablement se comporter comme ils l'ont déjà fait. Une telle approche aide les agents à se mettre d'accord.

3.2.4 Conclusion

L'approche permise par le formalisme des MMDP est intéressante pour mettre en valeur ce qu'est un problème de coordination quand il ne s'agit que de mettre d'accord des agents. Les hypothèses faites dans ce cadre de travail sont néanmoins très contraignantes, puisqu'il est supposé que tout agent a une observation complète du système, et peut donc aisément adopter une fonction de récompense commune avec ses camarades (ce qui serait plus difficile avec des observations partielles).

3.3 DEC-POMDP

[Bernstein *et al.*, 2000; 2002]

Progressant vers les situations qui nous intéressent (voir introduction de ce chapitre), nous allons maintenant examiner des travaux tenant compte du fait que les agents peuvent avoir des observations partielles. Nous commençons par voir ici les résultats d'études de complexité présentés dans [Bernstein *et al.*, 2002] (dans des situations à horizon fini).

Les auteurs se préoccupent, plus précisément, de la planification centralisée (récompense commune, comme dans le cas des MMDP) pour des agents distribués ayant accès à des informations incomplètes. Les auteurs ont étendu le modèle du PDMPO pour permettre à des agents au sein d'un groupe de percevoir chacun leur observation et de fonder leur décision sur ces observations. La fonction de transition (T) ainsi que la fonction de récompense (R) restent jointes, c'est-à-dire qu'elles dépendent des actions de tous les agents (ce qui reste similaire aux MMDPs). Ils ont appelé ce modèle "DEC-POMDP" pour *DECentralized Partially Observable Markov Decision Process*. Des modèles très proches se retrouve dans la littérature : dans [Peshkin *et al.*, 2000] par exemple, sous la dénomination de *POIPSG* (jeu stochastique à gains identiques partiellement observables).

3.3.1 Formalisme

Le DEC-POMDP permet un contrôle décentralisé. Dans ce modèle à chaque pas de temps, chaque agent reçoit une observation locale et choisit une action. La fonction de transition ainsi que la fonction de récompense dépendent du vecteur des actions de tous les agents.

Définition 7 (DEC-POMDP)
Un DEC-POMDP pour deux agents est défini par $\langle S, A_1, A_2, P, R, \Omega_1, \Omega_2, O, T, K\rangle$ *où :*
- S *est un ensemble fini d'états.*
- A_1 *et* A_2 *sont des ensembles finis d'actions pour respectivement les agents* 1 *et* 2.
- T *est la fonction de transition.* $T(s, a_1, a_2, s')$ *représente la probabilité de transition de l'état* s *à* s' *quand les agents* 1 *et* 2 *effectuent respectivement les actions* a_1 *et* a_2.
- R *est la fonction de récompense.*
- Ω_1 *et* Ω_2 *sont les ensembles finis des observations.*
- O *est la fonction d'observation.* $O(s, a_1, a_2, s', o_1, o_2)$ *représente la probabilité que les observations respectives des agents* 1 *et* 2 *soient* o_1 *et* o_2 *pour la transition* $(s, a_1, a_2) \to s'$.
- $\mathcal{T}$ *est un entier positif représentant l'horizon (on se place dans un cadre à horizon fini).*
- K *est un réel représentant le seuil de valeur. Le problème de décision posé est alors de savoir s'il existe une politique qui permette de dépasser ce seuil.*

3.3.2 Résultats

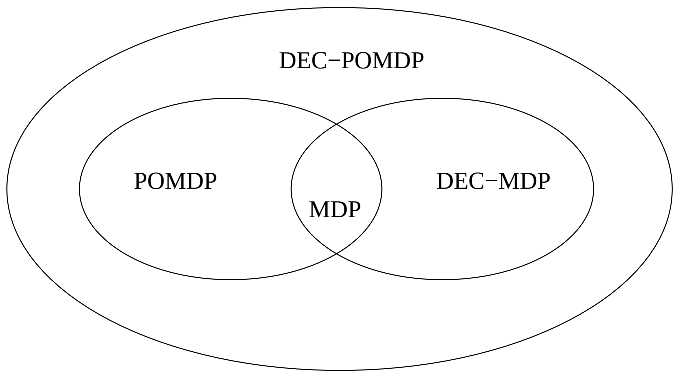


FIG. 3.7 – Relations entre les différents modèles.

Un DEC-POMDP généralise un PDMPO en modélisant un groupe d'agents qui, ensemble, ne parviennent pas à observer complètement le système. Dans le cas où les observations jointes des agents permettent une information complète, on parle de DEC-MDP (attention : dans un DEC-MDP, chaque agent peut n'avoir qu'une observation partielle du système).

Dans cet article, Bernstein et al. ont montré que le DEC-POMDP et le DEC-MDP à horizon fini sont NEXP-difficiles pour un nombre d'agents $n \geq 2$. Lorsque le problème est limité à un horizon plus petit que le nombre d'états, les problèmes sont NEXP-complet. Tout comme pour le PDMPO, dans le cas d'un horizon infini, le DEC-POMDP et le DEC-MDP sont non décidables pour certains critères d'optimalité [Madani *et al.*, 1999].

3.3.3 Conclusion

Le modèle DEC-POMDP formalise le contrôle décentralisé d'un système multi-agents : la fonction d'observation est individuelle. En cela, il se rapproche d'une situation réaliste d'un groupe d'agents autonomes. En revanche, comme pour les MMDP, la fonction de récompense est globale, ce qui ne permettrait qu'une conception centralisée en pratique (à moins de supposer un mécanisme d'assignation de récompenses global).

Les résultats théoriques obtenus nous révèlent qu'il est "impossible" de résoudre un DEC-POMDP dès que le nombre d'agents est supérieur ou égal à 2. Résoudre un problème de type DEC-POMDP ne sera dans ce cas possible qu'en utilisant des méthodes approchées. Une difficulté persistante est alors de qualifier la qualité des solutions obtenues.

3.4 Descentes de gradient

[Peshkin *et al.*, 2000; Bartlett et Baxter, 1999]

En section 2.3.3.2 a été présenté un algorithme de descente de gradient pour faire une recherche directe de politique. Outre ce travail [Baxter et Bartlett, 2001; Baxter *et al.*, 2001], d'autres se sont aussi intéressés à l'utilisation de descentes de gradient dans le même cadre de processus de décision markoviens partiellement observables, comme c'est le cas par exemple des travaux présentés dans [Meuleau *et al.*, 1999; Peshkin *et al.*, 1999].

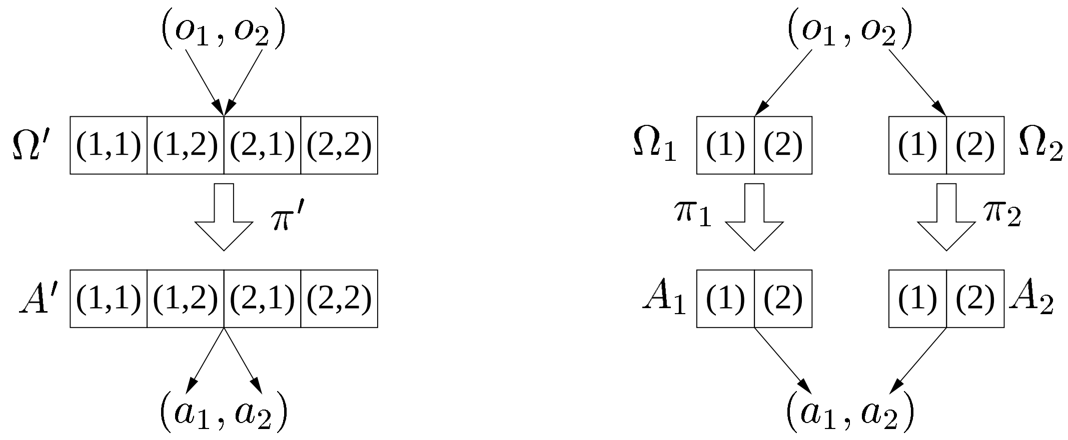
3.4.1 Contrôle centralisé d'un DEC-POMDP

Une descente de gradient, sauf forme particulière, permet de définir l'ensemble de paramètres[27] de manière à en associer un par paire observation-action ($\theta_{o,a}$), comme nous l'avons fait dans le chapitre précédent. Avec $|\Omega| * |\mathcal{A}|$ paramètres, on peut ainsi gérer tous les degrés de liberté d'une politique (on pourrait en fait se limiter à $|\Omega| * (|\mathcal{A}| - 1)$ paramètres).

Dans le cas d'un DEC-POMDP (pour reprendre le formalisme vu précédemment), on peut concevoir un système de contrôle *centralisé* en ramenant donc le problème d'un groupe d'agents à un agent seul, comme le montre la figure 3.8-a. Les observations et actions disponibles pour cet agent unique sont alors les observations et actions jointes ($\Omega' = \Omega_1 \times \cdots \times \Omega_n$ et $A' = A_1 \times \cdots \times A_n$). Le nombre de paramètres d'une descente de gradient devient (suivant toujours le même choix) : $|\Omega'| * |A'|$.

Remarque : ici, toute action a_i du “sous-agent” i dépend des perceptions de tous les agents.

[27] Rappel : dans ce cadre, la politique n'a pas de forme contrainte a priori. C'est au concepteur de la définir sous une forme paramétrée (en choisissant l'ensemble de paramètres).



a- Contrôle centralisé :
- Ω' espace d'observation ($|\Omega'| \leq \prod_i |\Omega_i|$).
- A' espace d'action ($|A'| = \prod_i |A_i|$).
- π' politique ($|\Omega'| * |A'|$ paramètres).

b- Contrôle décentralisé :
- Ω_i espace d'observation de i.
- A_i espace d'action de i.
- π_i politique de i ($|\Omega_i| * |A_i|$ paramètres).

FIG. 3.8 – Schéma de fonctionnement d'un DEC-POMDP composé de deux agents : a- centralisé et b- décentralisé.

3.4.2 Contrôle décentralisé d'un DEC-POMDP

Mais le même type d'algorithmes permet de concevoir, de manière centralisée, des agents indépendants. Il suffit pour cela de définir comme ensemble de paramètres du gradient l'union des ensembles de paramètres dédiés à chaque agent. En notant $\theta_{i,o,a}$ un paramètre pour la paire observation-action (o, a) de l'agent i, l'ensemble de paramètres global est :

$$\begin{aligned} \theta' &= \{\, \theta_{i,o,a},\ i \in [1..n],\ (o, a) \in \Omega_i \times \mathcal{A}_i \,\} \\ &= \bigcup_i \{\, \theta_{i,o,a},\ (o, a) \in \Omega_i \times \mathcal{A}_i \,\}. \end{aligned}$$

Un schéma de principe est montré en figure 3.8-b, dans lequel chaque politique π_i est définie par un sous-ensemble $\theta_i = \{\, \theta_{i,o,a},\ (o, a) \in \Omega_i \times \mathcal{A}_i \,\}$ des paramètres.

Remarque : ici, toute action a_i de l'agent i dépend uniquement des perceptions de cet agent.

On a alors un algorithme qui permet d'apprendre les $\sum_i |\Omega_i| * |A_i|$ paramètres du groupe d'agents, approche employée dans [Bartlett et Baxter, 1999] ou [Peshkin *et al.*, 2000] par exemple. Elle permet d'obtenir des agents complètement coopératifs, mais qui ne sont plus capables de s'adapter une fois en situation décentralisée (sauf si on continue à leur fournir un signal de renforcement commun).

3.4.3 Conclusion

La principale différence entre l'apprentissage pour un contrôle centralisé et un contrôle décentralisé est que, dans le premier cas, une quelconque décision (au niveau d'un "sous-agent") dépend de l'ensemble des observations alors que, dans le second cas, une telle décision n'est liée qu'à l'observation du dit sous-agent.

Cette différence met en relief le fait que, avec un contrôle décentralisé, les agents ont accès à beaucoup moins d'informations pour prendre leurs décisions. Cette méthode permet toutefois de concevoir un groupe d'agents coopérants d'une manière des plus efficaces (aux problèmes d'optima locaux près) si une phase de conception peut ainsi être séparée de l'utilisation du système multi-agents obtenu.

3.5 COIN

[Tumer et Wolpert, 2000][Wolpert *et al.*, 1999]

3.5.1 Introduction

Jusqu'à présent, les différents travaux vus sur le problème de conception d'agents coopérants supposaient une récompense commune à tous les agents, donc une forme de centralisation, alors que nous cherchons autant que faire se peut à nous affranchir de telles contraintes. Le problème qui se pose donc est de savoir quelles fonctions de récompense liées à chacun des agents vont permettre une réelle coopération comme c'est le cas quand une main extérieure donne la même récompense à tous les agents[28].

Dans le but d'étudier ce problème précis, le cadre des COIN (pour COllective INtelligence) a été défini et utilisé en premier lieu dans [Tumer et Wolpert, 2000; Wolpert *et al.*, 1999].

3.5.2 "Tragedy of the Commons"

Le principe qu'il faut combattre ici est connu comme étant le paradoxe de Braess [Bass, 1992]. Pour le présenter, nous nous servirons de l'exemple de la "tragédie des communs", tiré d'un pamphlet du mathématicien amateur William Forster Lloyd (1794-1852) [Lloyd, 1833].

> Imaginez-vous une pâture ouverte à tous. Il faut s'attendre à ce que chaque bouvier[29] essaiera de garder autant de bétail que possible sur ces communs. Un tel arrangment peut fonctionner de manière plutôt satisfaisante pendant des siècles à cause de guerres tribales, de braconnages, et de maladies qui gardent le nombre d'hommes comme de bêtes bien en-deça de la capacité d'accueil du terrain. Finalement, toutefois, vient le jour des règlements, c'est-à-dire le jour où le but longtemps visé de stabilité sociale devient une réalité. A ce moment, la logique inhérente des communs génère impitoyablement une tragédie.
>
> En tant qu'être rationnel, chaque bouvier cherche à maximiser son gain. Explicitement ou implicitement, plus ou moins consciemment, il se demande "Quel est pour moi l'utilité d'ajouter un animal de plus à mon troupeau ?" Cette utilité a un composant négatif et un positif.
>
> 1. Le composant positif est fonction de l'ajout d'un animal. Puisque le bouvier reçoit tout le montant de la vente de cet animal additionnel, l'utilité positive est à peu près de $+1$.
> 2. Le composant négatif est fonction du sur-broutage additionnel créé par un animal supplémentaire. Puisque, toutefois, les effets de ce sur-broutage sont partagés par tous les bouviers, l'utilité négative pour un quelconque bouvier décideur n'est qu'une fraction de -1.
>
> En additionnant ensemble les utilités partielles des composants, le bouvier rationnel conclut que la seule marche à suivre sensée pour lui est d'ajouter un autre animal à

[28] ***Note :*** l'usage d'une récompense commune peut aussi rendre l'apprentissage difficile, puisqu'un agent ne saura pas réellement dans quelle mesure son comportement est lié à la récompense qu'il reçoit.

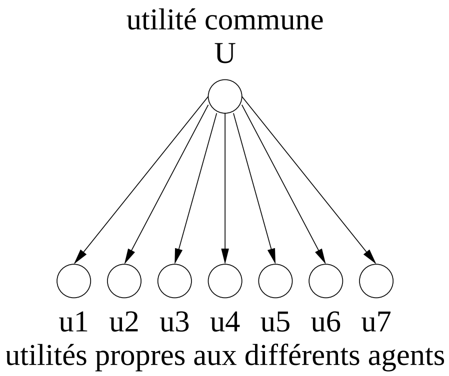
[29] conducteur de bestiaux

son troupeau. Et un autre... Mais c'est cette conclusion que va atteindre tout bouvier partageant ces terres.
Là est la tragédie. Chaque homme est bloqué dans un système qui le force à agrandir son troupeau sans limite – dans un monde qui est limité. La ruine est la destination vers laquelle tous les hommes se ruent, chacun poursuivant son propre intérêt dans une société qui croît à la liberté des communs. La liberté dans des communs amène la ruine de tous.

Le lecteur intéressé pourra lui-même chercher comment cette situation peut se transcrire si l'on considère une population qui a le choix entre l'usage de transports en commun ou de véhicules personnels. Ce qui est important, c'est de voir qu'ici l'intérêt de chacun va contre l'intérêt de tous.

3.5.3 Approche proposée

Un COIN, tel que présenté par ses créateurs, est un système multi-agents sans communication ni commande centralisée, et pour lequel une fonction d'utilité "du monde" est définie (donc commune à tout le groupe d'agents, comme pour les MMDP et DEC-POMDP vus précédemment). L'idée de Tumer et Wolpert est de faire en sorte que cette utilité commune soit décomposable en utilités propres aux différents agents (utilités auxquelles ils aient accès), de façon à ce que les variations des utilités de chacun amènent les mêmes variations à l'utilité commune.



Partant d'une utilité commune, l'objectif est de trouver les utilités propres aux différents agents et qui garantissent la coopération. Si le gain d'un agent augmente, le gain commun en fait autant.

FIG. 3.9 – Principe de distribution de l'utilité "du monde".

Ce principe assure que l'amélioration de la politique de chaque agent mène à une amélioration globale du comportement de groupe. On peut ainsi être certain d'atteindre un optimum au niveau du groupe tout en n'étant pas obligé d'avoir une récompense commune à tous (ce qui requiert une centralisation). Néanmoins, la mise en application présentée dans [Tumer et Wolpert, 2000] ne concerne qu'un problème d'optimisation de routage, et ne dit ni dans quels cas on pourra mettre en œuvre cette approche, ni comment.

On peut trouver dans [Wolpert *et al.*, 1999] une discussion sur les problèmes dans lesquels on peut appliquer la "méthode COIN". Pour résumer leurs résultats, Wolpert et Tumer en sont arrivés à montrer qu'une condition suffisante est que le système multi-agents puisse être décomposé en systèmes multi-agents indépendants (...).

3.5.4 Conclusion

L'analyse de départ faite dans ces travaux sur les COINs est intéressante dans la mesure où elle exprime assez clairement (à travers la tragédie des communs et le paradoxe de Braess) le problème qui apparaît quand on veut une coopération globale au niveau d'un groupe alors que les objectifs des éléments du groupe sont individuels. Mais, sauf nouveaux résultats, il ne semble pas encore évident de pouvoir automatiser la conception de COINs, puisque le problème posé n'est résolu que dans quelques cas bien particuliers.

3.6 Empathie

[Chadès, 2003]

3.6.1 Introduction

Pour l'instant, la seule approche "avec modèle" que l'on ait rencontré pour la conception de SMA coopératifs est l'algorithme *value iteration* avec extension dans le cadre des MMDP. Et encore, la phase de coordination randomisée (si l'on suit cette approche sans protocole ni communication) correspond à un apprentissage par essais-erreurs tels que ceux des approches sans modèle.

Partant d'une idée similaire, les travaux autour de la notion d'*empathie*, présentés dans [Chadès, 2003], cherchent à n'employer qu'une approche "planification" (sans essais-erreurs) pour réaliser des agents coopérants. Cette dernière présentation nous permettra de voir des algorithmes pratiques reposant sur une forme de modélisation d'autrui.

3.6.2 Principe

Dans le domaine de la psychologie, l'empathie se définit comme l'habileté à percevoir, à identifier et à comprendre les sentiments ou les émotions d'une autre personne tout en maintenant une distance affective par rapport à cette dernière. En psychologie sociale, le terme empathie, dans un sens élargi, désigne aussi la capacité d'acquérir une certaine connaissance d'autrui, de se mettre à sa place.

Le principe de "raisonner" en fonction de ce que l'on sait des autres a déjà été rencontré dès la section 3.1.1, puisque les travaux de la théorie des jeux intègrent pour la plupart la présence d'autres joueurs ayant leurs buts propres. Dans le cas présent, c'est pour deux raisons différentes que la propriété d'empathie des agents est considérée :

- En conception centralisée, elle permet de travailler par étapes sur des sous-espaces de l'espace des politiques jointes, lequel risque d'être très grand ($|\mathcal{S}| \times |A_1| \times \cdots \times |A_n|$).
- En conception décentralisée, elle permet à chaque agent d'améliorer (toujours par étapes) sa politique sur la base des politiques des autres agents.

3.6.3 Théorie

L'algorithme proposé consiste à fixer d'abord un sous-ensemble B_1 des agents pour pouvoir appliquer l'algorithme *value iteration* sur le sous-ensemble restant A_1(dans un système dont la fonction de transition T_1 dépend donc du comportement des agents de B_1, puisqu'ils appartiennent à l'environnement), puis d'itérer en choisissant un deuxième sous-ensemble fixe B_2, puis un troisième... En procédant ainsi, la fonction de valeur commune augmente à chaque étape, ce qui assure la convergence de l'algorithme.

Si les groupes A_i sont de taille 1 au plus, le point atteint sera un équilibre de Nash classique : aucun agent ne pourra plus, par un changement de sa seule politique, améliorer son gain (et donc ici celui de ses congénères). De manière générale, si les groupes A_i sont de taille k au plus, on pourra parler de convergence vers un équilibre de Nash de "degré k" (aucun groupe de k agents ne peut améliorer le gain du groupe).

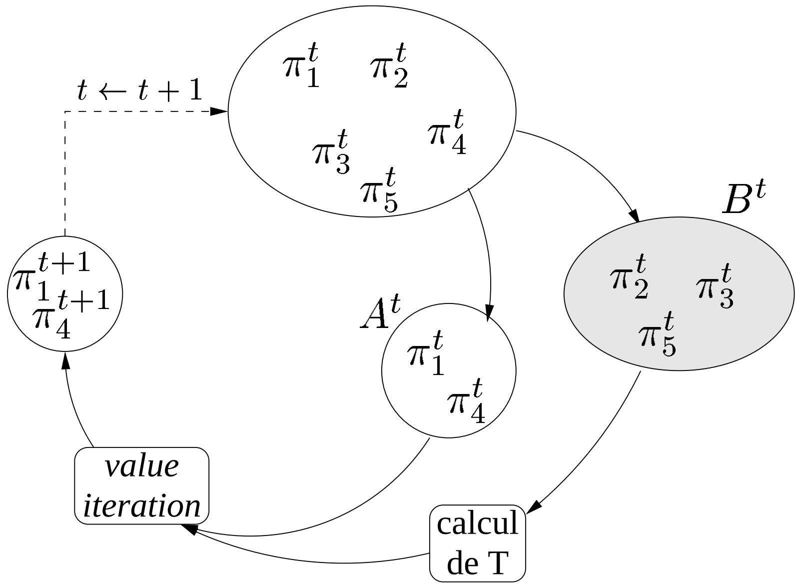


FIG. 3.10 – Utilisation du principe d'empathie.
Dans cet exemple, sur 5 politiques, 3 sont fixées (sous-ensemble B^t) et permettent de calculer la fonction de transition T de ce qu'est l'environnement du point de vue du sous-ensemble restant A^t. On peut alors appliquer l'algorithme *value iteration* pour améliorer les 2 politiques de A^t.

3.6.4 Pratique

Le principe théorique qui vient d'être décrit permet a priori de simplifier la conception d'un SMA en travaillant par sous-groupes d'agents mais ce, au prix d'une solution sous-optimale (l'équilibre de Nash de "degré k"). Malheureusement, on retombe sur la problématique évoquée en section 3.2 dès qu'il s'agit de mettre au point les comportements coopératifs de plus d'un agent. Mais si la conception est centralisée ou si un protocole permet de résoudre les conflits possibles, ces problèmes ne se posent plus.

Les différents algorithmes pratiques proposés dans [Chadès, 2003] correspondent à des situations particulières (perceptions subjectives, planification décentralisée d'une politique commune, récompenses distribuées) dans lesquelles on perd les propriétés de convergence présentées. Nous ne développerons donc pas ici ces algorithmes.

3.6.5 Conclusion

L'utilisation du principe d'empathie montre encore une fois l'intérêt qu'il y a à modéliser les autres agents. Il est en plus utilisé pour réduire la complexité de conception d'un groupe complet d'agents, mais au prix d'une efficacité moindre. Cependant, l'approche "avec modèle", et en supposant la propriété de Markov vérifiée, ne correspond pas à nos hypothèses de travail.

3.7 Bilan

Les différents travaux présentés dans ce chapitre donnent finalement un bon aperçu de ce qu'est le problème de l'apprentissage par renforcement dans un système multi-agents. Quelques aspects importants qu'on peut retenir sont :

- l'importance de la répartition d'une récompense commune en récompenses individuelles de manière adéquate, point sans solution simple,
- les difficultés de coordination et les situations d'équilibres sous-optimaux qui empêchent le groupe d'atteindre la meilleure politique jointe,
- le rôle de la modélisation des autres agents (par des approches telles que l'empathie), laquelle permet une meilleure adaptation au reste du groupe,
- et finalement la complexité générale du problème que décrit un DEC-POMDP.

Tout ceci rend la réalisation d'un SMA par apprentissage par renforcement particulièrement difficile. Pour l'instant, seules certaines approches centralisées permettent de garantir des agents dont la politique jointe est localement optimale (c'est le cas des descentes de gradient).

Souhaitant éviter une phase de conception centralisée de ce type, les méthodes proposées dans le chapitre suivant se contenteront d'*aider l'apprentissage* par des moyens heuristiques dont nous évaluerons la qualité. De plus, souhaitant pouvoir traîter des problèmes assez réalistes, on ne fera pas l'hypothèse souvent rencontrée de perceptions totales, d'où le choix d'un algorithme de recherche directe de politique pour effectuer des apprentissages individuels (la montée de gradient de Baxter vue en section 2.3.3.2).

Chapitre 4

Apprentissage progressif

Sommaire

[30] Regarde maman ! Sans les p'tites roues !

Comme le chapitre 3 nous a permis de le voir, la conception d'agents par des approches de type apprentissage par renforcement pose diverses difficultés. Elle s'avère même particulièrement difficile avec des agents indépendants les uns des autres (approche décentralisée), entre autres du fait que l'on ne sait pas passer de récompenses collectives à des récompenses individuelles.

Nous allons nous intéresser dans le présent chapitre à des méthodes d'aide à l'apprentissage fondées sur une progression de la tâche à accomplir. Elles nous font hélas retomber dans des approches proche du supervisé, ce qui est contraire à notre vœux d'autonomie des agents, mais améliorent de manière notable l'efficacité de l'apprentissage, ce qui justifie leur intérêt.

Des travaux apparentés aux notres seront tout d'abord présentés en section 4.1. Les sections 4.2 et 4.3 qui suivent s'intéresseront aux deux aspects de la méthode que nous proposons : un progrès de la tâche à accomplir en complexité, et un progrès en nombre d'agents présents.

4.1 Travaux précédents

L'idée de faire un apprentissage progressif est connue généralement sous le terme anglais de *"shaping"*, que l'on peut traduire par *"apprentissage progressif"*. La section 4.1.1 qui suit introduit cette notion à travers un court historique. Les deux sections suivantes, pour leur part, présentent des travaux utilisant des approches d'apprentissage progressif en apprentissage machine. Une courte dernière section nous permettra d'introduire les méthodes proposées dans ce domaine par la présente thèse.

4.1.1 Historique

Pour cet historique, nous nous contentons de traduire ici une courte introduction tirée de [Randløv et Alstrøm, 1998] :

> L'idée d'apprendre de manière "incrémentale", qui est empruntée à la psychologie comportementale, consiste à donner à l'agent apprenant une série de problèmes relativement simples et évoluant vers le problème plus difficile auquel va notre intérêt [Sutton et Barto, 1998]. Le terme vient des travaux du psychologue Skinner [Skinner, 1938], qui en étudia les effets sur des animaux, en particulier des pigeons et des rats.
>
> Pour entraîner un animal à produire un certain comportement, l'entraîneur doit trouver quelles sous-tâches constituent une approximation du comportement désiré, et comment elles devraient être renforcées [Staddon, 1983]. En récompensant des approximations successives du comportement désiré, on peut amener des pigeons à picorer en un endroit précis [Skinner, 1953, p. 93], des chevaux à faire des tours de cirque tels que sembler reconnaître des drapeaux de nations ou des nombres et effectuer des calculs [Jørgensen, 1962, pp. 137–139], et des cochons à accomplir des actes complexes tels que déjeuner à table et passer l'aspirateur [Atkinson *et al.*, 1996, p. 242]. Staddon note que l'éducation humaine elle aussi est fondée sur un tel processus d'apprentissage progressif si l'on considère que le comportement inclue la "compréhension" [Staddon, 1983, p. 458].

Comme les travaux sur le conditionnement ont inspiré l'apprentissage par renforcement, l'étude de l'apprentissage progressif a, elle aussi, mené à différents développements en intelligence artificielle. Nous allons voir maintenant des travaux qui entrent dans ce cadre, ou du moins s'en rapprochent fortement.

4.1.2 Look, Ma ! No training wheels ![31]

Les articles [Randløv et Alstrøm, 1998; Randløv, 2000] peuvent constituer un bon point de départ pour explorer le domaine de l'apprentissage progressif artificiel (l'introduction historique que nous venons d'en tirer le prouve).

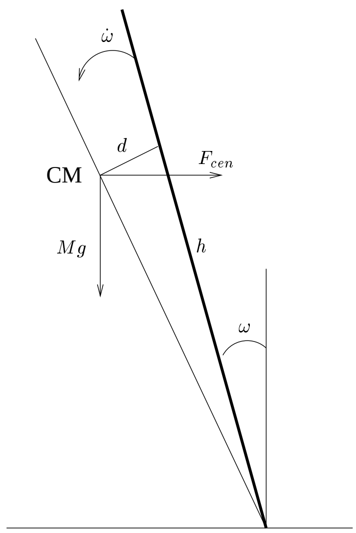
Le principal problème qui illustre les travaux de Randløv est celui de l'apprentissage du vélo, problème déjà difficile chez l'humain, et donc des plus parlant. Nous allons d'abord préciser le modèle utilisé pour ce problème, avant de regarder les méthodes discutées dans les deux papiers.

Note : Tout ce qui est discuté ci-après suppose l'utilisation de processus de décision markoviens, et donc de la classique espérance de gain décompté (avec $\gamma \in [0, 1)$ quand l'horizon est infini).

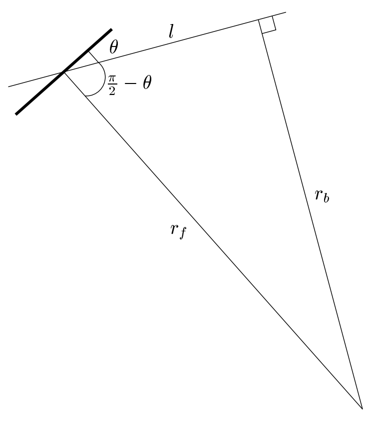
Problème posé

L'objectif est de réussir à garder l'équilibre sur la bicyclette et à atteindre un point donné. L'agent contrôle le couple à appliquer au guidon ($T \in \{-2\text{N}, 0\text{N}, +2\text{N}\}$) et le déplacement du centre de masse par rapport au plan de la bicyclette ($d \in \{-2\text{cm}, 0\text{cm}, +2\text{cm}\}$), d'où un total de 9 actions (dont l'effet est bruité). Les perceptions sont aussi discrétisées (sur 5 ou 7 intervalles chacune), et concernent :
- θ l'angle du guidon avec sa position normale,
- $\dot{\theta}$ la vitesse angulaire de cet angle,
- ω l'angle entre la bicyclette et la verticale,
- $\dot{\omega}$ la vitesse angulaire associée, et
- $\ddot{\omega}$ l'accélération angulaire associée.

Pour plus de détails voir [Randløv et Alstrøm, 1998], qui fait même une description du modèle physique employé. Les figures 4.1 a) et b) montrent comment on peut schématiser une bicyclette vue de face et de dessus.



a) Vue de face. La ligne épaisse représente la bicyclette. CM est le centre de masse de l'ensemble bicyclette+cycliste.



b) Vue de dessus. La ligne épaisse représente la roue avant (l décrit la longueur du vélo).

FIG. 4.1 – Deux représentations schématiques d'une bicyclette, vue a) de face et b) de dessus.

[31] Regarde maman ! Sans les p'tites roues !

C'est donc entre autres sur cet exemple que Randløv montre que l'apprentissage progressif peut être accompli de différentes manières. En l'occurrence, deux approches sont examinées ici, approches que nous allons maintenant voir dans les deux sections suivantes.

Changement du signal de récompense

Dès [Randløv et Alstrøm, 1998], la forme d'apprentissage progressif proposée s'oriente sur une modification du signal de renforcement. On peut voir deux aspects à la définition de la fonction de récompense :

1. On distingue des récompenses liées à l'objectif de garder l'équilibre et d'autres conduisant à trouver la zone but. Les secondes ne doivent pas être trop marquées, sinon l'agent cherche à rejoindre le but sans maîtrise de l'équilibre du vélo.
2. Pour ce qui est d'atteindre le but, le renforcement est positif quand on se rapproche et négatif en cas d'éloignement. Cela permet de ne pas attendre de tomber par hasard sur le but pour commencer à apprendre quelque chose et rend ainsi la tâche réalisable.

Les expérimentations effectuées confirment l'intérêt des idées proposées. On note toutefois qu'il peut être délicat de trouver une fonction de récompense correcte : de premiers essais ne pénalisaient pas l'éloignement (du but), ce qui a conduit à un cycliste tournant autour du point de départ, accumulant des bons points à chaque tour. On retrouve là le problème illustré par un robot-aspirateur sur la figure 1.14 (section 1.3).

[Ng *et al.*, 1999] s'est précisément intéressé aux modifications qui peuvent être apportées à une fonction de récompense sans que la politique optimale en soit modifiée. Dans le formalisme utilisé, la nouvelle récompense $R'(s,a,s')$ s'écrit à partir de l'ancienne : $R(s,a,s') + F(s,a,s')$, où F est donc une "fonction de récompense progressive".

Le principal résultat obtenu est que, dans le cas général, une fonction de type "différence de potentiel" entre deux états ($F(s,a,s') = \gamma\Phi(s') - \Phi(s)$, pour un γ fixé) garantit la conservation de la politique optimale. Cette règle est utilisée dans la nouvelle modification du signal de récompense proposée par [Randløv, 2000] pour garder l'équilibre : $\Phi(s) = -0.1\omega^2$. On obtient sans difficulté une nette amélioration de la vitesse d'apprentissage.

Changement de la physique du problème

Mais [Randløv, 2000] réoriente son intérêt vers un autre moyen de mener un apprentissage progressif. Il ne s'agit plus de manipuler R, mais T, ou en d'autres termes de changer la physique du problème. Si c'est à un problème réel (de notre monde physique) que l'agent est confronté, le principe de l'approche est alors de passer par des modèles successifs qui tendent vers un modèle réaliste, mais qui permettent de guider et donc d'accélérer l'apprentissage.

Comme [Ng *et al.*, 1999] l'avait fait pour l'utilisation d'un signal de renforcement modifié, un résultat théorique est apporté par Randløv pour cette seconde approche. En effet, il est démontré dans [Randløv, 2000] que si la suite de fonctions de transitions T_k tend vers la fonction réelle T, alors la suite de fonctions de Q-valeurs optimales Q_k^* tend vers Q^* (il faut aussi $\gamma < 1$, même pour un horizon fini) :

$$\text{Si } T_k \to T \quad \text{Alors} \quad Q_k^* \to Q^* \tag{4.1}$$

Pour revenir à notre problème de cycliste débutant, ce principe est mis en œuvre d'une manière assez intuitive en mettant des petites roues sur les côtés du vélo, et en les relevant progressivement au cours de l'apprentissage jusqu'à ce qu'elles ne servent plus à rien. Cela correspond bien à une modification de la physique du problème, avec une évolution vers la situation réelle. C'est fait assez simplement en empêchant l'angle ω du vélo de dépasser une limite liée aux petites roues.

Ici encore, les résultats expérimentaux sont satisfaisants. Mais sur un autre problème, celui de la "voiture sur la montagne" [Moore, 1991] présenté sur la figure 4.2, des difficultés apparaissent. En utilisant une montagne de plus en plus haute, la voiture n'a d'abord pas besoin d'élan, et apprend alors assez bien. Mais la phase correspondant aux cas où l'élan est d'abord utile (parce qu'améliore les performances), voire plus tard nécessaire, s'avère difficile. Il faut pendant cette phase là complètement changer ses habitudes : décider de partir en arrière pour mieux se relancer, ce qui est en rupture avec les politiques jusque là employées.

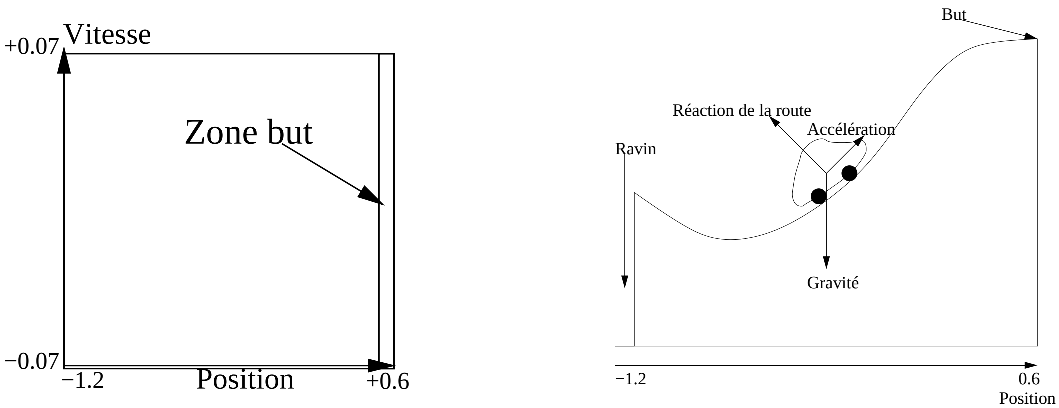


FIG. 4.2 – Problème de la voiture sur la montagne (figure gracieusement prêtée par Bruno Scherrer [Scherrer, 2003]).

> Une voiture peu puissante doit atteindre le haut d'une montagne. Pour ce faire, elle doit osciller dans le bassin et utiliser la gravité afin de gagner assez d'énergie cinétique pour gravir la montagne. Elle doit faire attention à ne pas prendre trop de vitesse sinon elle risque de tomber dans le ravin. La dynamique de ce problème peut être décrite dans l'espace des phases (position,vitesse). L'objet de ce problème de contrôle est de trouver un chemin jusqu'à la zone but dans cet espace des phases en contrôlant l'accélération du véhicule.

Conclusion

S'il est prouvé dans les deux cas présentés (problèmes du vélo et de la voiture) qu'on a une convergence vers une solution optimale, aucun résultat ne dit quelles modifications du signal de récompense ou de la physique du problème permettent d'accompagner les progrès de l'agent apprenant. Il reste donc a priori nécessaire d'apporter des connaissances du concepteur, contrairement au souhait d'autonomie exprimé en section 1.1.6.

Dans ces papiers est aussi évoqué le fait que d'autres voies sont à explorer dans le domaine de l'apprentissage progressif, telles que la réduction de l'espace d'états (ce qui peut être relié aux travaux sur la discrétisation automatique de [Moore, 1991; Munos et Moore, 2002; Scherrer, 2003] par exemple), ou l'utilisation d'un coefficient de décompte γ variable.

Pour conclure, on notera le lien fait avec l'approche qu'on pourrait dire "d'auto-émulation" ou d'autodidactisme (*self-play*) employée pour l'apprentissage du Backgammon dans [Tesauro, 1992; 1994]. En effet, dans cet exemple l'agent apprend en jouant contre lui-même. Ainsi, la tâche est plutôt aisée au début, puisque son adversaire n'est pas très fort, mais est progressivement rendue plus ardue puisqu'il s'améliore au fur et à mesure des parties.

Il est à noter que, à l'inverse de ce cas compétitif, des agents coopérants vont d'abord avoir du mal à se coordonner, puis que l'apprentissage va s'accélérer (comme dans un apprentissage mono-agent où l'agent commence à trouver le chemin à suivre).

4.1.3 Apprendre à partir de missions simples

Les travaux que nous allons maintenant citer sont ceux présentés dans [Asada et al, 1996]. Il n'y est pas fait référence explicitement à d'autres travaux du domaine de l'apprentissage progressif, mais il semble sans équivoque qu'ils en fassent partie. Si nous nous y intéressons, c'est qu'ils ressemblent beaucoup à notre propre approche, comme on pourra le constater ultérieurement.

Problème posé

Dans cet article est donc proposée une méthode pour qu'un robot "footballeur" apprenne à marquer des buts (voir figure 4.3). Il s'agit d'une des premières tentatives (réussies) d'apprentissage par renforcement avec un vrai robot, et ce, en utilisant comme unique source d'information son système de vision embarquée.

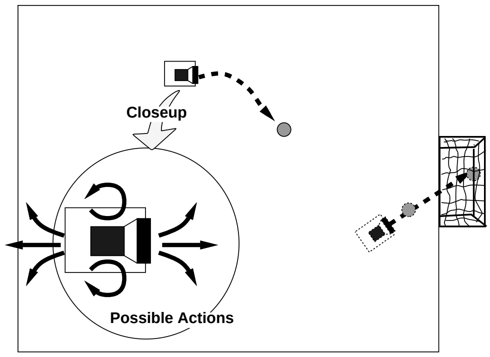


Fig. 4.3 – Le problème posé : le robot doit apprendre à marquer des buts. (image tirée de [Asada et al, 1996])

PDM

Le robot ne connaît pas de modèle (physique ou autre) de son environnement. Il est supposé que ses perceptions seront suffisantes pour que la propriété de Markov soit vérifiée, et donc qu'elles définissent des états (au sens d'un PDM, voir sur le sujet la section 2.2). Ainsi, on peut utiliser le classique Q-learning comme algorithme d'apprentissage en ligne.

L'utilisation d'un PDM en robotique pose quelques problèmes au concepteur. Dans ce cas précis, voici les points discutés par les auteurs (en simplifiant un peu leurs propos) :

- Le **temps** est a priori continu. En l'occurrence, on le discrétise en utilisant des perceptions reçues à la fréquence de traîtement de la caméra.
- L'**espace des observations** est, lui, quasiment continu (les images possibles sont en nombre fini, mais très grand). Le robot ne pouvant filmer que la balle ou le but, il a été décidé de classer les images possibles de la balle en 9 situations (selon sa position et sa taille à l'écran), et celle du but en 27 situations (selon sa position, sa taille et son orientation). Le tout est illustré sur la figure 4.4. On arrive à 319 états en tenant compte de cas dans lesquels l'un ou l'autre objet n'est pas visible.

– Les **actions** sont au nombre de 9, combinant les commandes AVANT, ARRIÈRE et STOP pour les moteurs droit et gauche (robot de type char). Ici encore, on pourrait avoir un contrôle plus fin, mais en rendant le problème plus difficile.
– La **récompense** de l'agent vaut simplement 1 quand un but est marqué, et 0 le reste du temps. Aucune notion de *progrès* vers l'état but n'est employée.

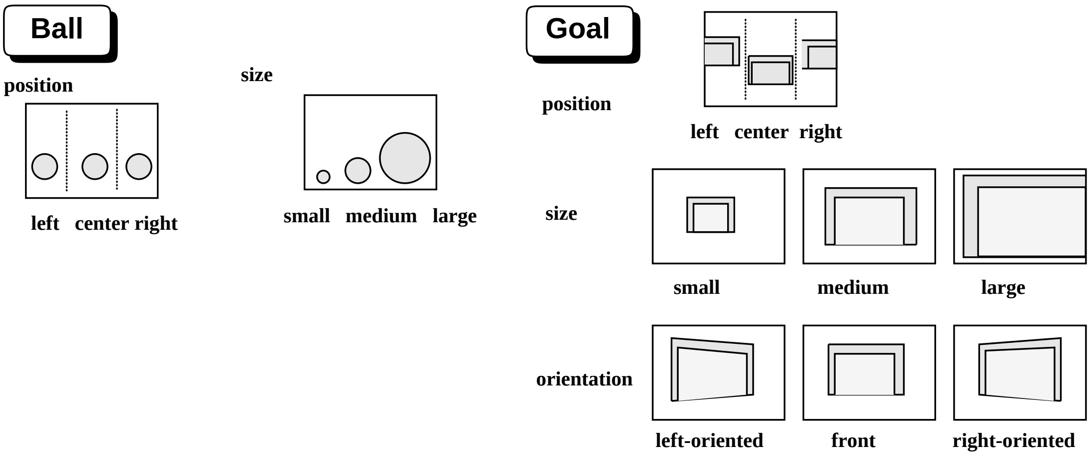


FIG. 4.4 – Les sous-états de la balle et du but.
(image tirée de [Asada et al, 1996])

Note : Les actions qui viennent d'être décrites amenant souvent à rester dans le même état, les variations dans les transitions sont très grandes. Il a donc été choisi de renommer ces actions des "actions primitives" et de ne considérer qu'une action n'a été accomplie que quand il y a changement d'état.

Learning from Easy Missions

Comme cela a été fait dans [Connell et Mahadevan, 1993], on peut améliorer la vitesse d'apprentissage en séparant la tâche complète en différentes parties : en l'occurrence trouver la balle, la déplacer, et tirer. Au contraire, la première approche choisie dans les travaux d'Asada auxquels nous nous intéressons ([Asada et al, 1996]) était monolithique, le robot débutant son apprentissage depuis des positions arbitraires. Malheureusement, la récompense en cas de but met beaucoup de temps à se propager dans de telles situations (on parle de *délai de renforcement*), ce qui rend cette approche irréaliste.

Pour éviter cette difficulté, un "programme d'entraînement" est défini de manière à ce que le robot apprenne dans des situations simples dans les premiers temps, puis dans des situations progressivement plus difficiles. Cette méthode est appelée *"Learning from Easy Missions"* (ou LEM, que l'on peut traduire par "apprendre à partir de missions simples).

En étudiant un exemple simplifié de PDM, les auteurs montrent que cette approche permet de passer d'une complexité temporelle *exponentielle* en k (où k est le nombre maximal d'étapes de leur PDM avant d'arriver au but) à une complexité *linéaire* (en k aussi). S'il n'est basé que sur un cas assez particulier, ce résultat illustre assez bien l'efficacité potentielle de ce principe LEM.

Il reste toutefois des difficultés à résoudre pour mettre en œuvre l'approche LEM. Ces difficultés sont de deux ordres :

- **Ordonnancement grossier des situations simples** : Une méthode simple afin de déterminer les “missions simples” auxquelles s'intéresser et l'ordre dans lequel les placer consiste à établir (si possible) une relation d'ordre sur les états du système. Dans le cas présent, l'espace d'observations continu et l'état objectif connu permettent de le faire, mais le problème reste ouvert dans les autres situations.
- **Quand se déplacer ?** : Un second problème d'implémentation est de décider à quel moment passer d'une situation initiale d'entraînement à une autre. La réponse apportée (dans le cas présent d'un espace d'état ordonné) est d'utiliser un critère mesurant la stabilisation de la table de Q-valeurs.

A partir de là, un algorithme a été défini et expérimenté. Les expériences menées se déroulent en deux parties :

- Une **simulation** permet d'abord d'appliquer l'algorithme et donc d'effectuer l'apprentissage de la politique. La comparaison avec un apprentissage *sans LEM* montre clairement l'intérêt de cette approche.
- Des **essais sur un vrai robot** donnent ensuite la possibilité d'évaluer la qualité réelle de la politique obtenue. Ici, le système de vision et la mécanique de la balle étaient insuffisament modélisés, ce qui amène à 60% de buts marqués seulement (accusant une baisse de 20% par rapport à la simulation).

Bilan

L'apprentissage à partir de missions simples s'avère être une méthode très efficace. La comparaison n'a pas été faite ici avec la décomposition de tâche de [Connell et Mahadevan, 1993], mais l'algorithme LEM a a priori l'avantage de ne pas demander d'aide manuelle (choix des sous-tâches). L'intervention humaine est en réalité indirecte : c'est le choix de perceptions “ordonnées” qui rend possible cette automatisation. De plus, lors de l'apprentissage, les informations acquises peuvent se “répandre” dans tout le PDM, alors que la diffusion est cloisonnée dans l'approche par décomposition.

L'approche LEM est aussi plus souple qu'un apprentissage supervisé, puisque très autonome (l'exploration se fait facilement). Sans le problème de construction d'un espace d'état adéquat, on aurait ici une méthode adaptée à une grande variété de problèmes, et ne nécessitant que l'utilisation d'un simulateur suffisament précis pour effectuer l'apprentissage d'une politique effectivement utilisable.

4.1.4 Bilan

Dans des situations où un apprentissage par renforcement classique se trouverait confronté à un espace d'exploration bien trop grand pour que la recherche d'une politique se fasse dans des temps raisonnables, des méthodes *progressives* telles que celles qui viennent d'être vues peuvent permettre une simplification (et donc une accélération) notable de la tâche. Comme les travaux d'Asada le montrent (mais on peut aussi citer [Colombetti *et al.*, 1996; Dorigo et Colombetti, 1997] et [Matarić, 1994; 1997]), de telles approches se montrent particulièrement intéressantes en robotique, le temps d'apprentissage étant un point crucial, et certaines expériences devant à tout prix être évitées.

Pour notre part, nous allons essayer d'appliquer de telles méthodes de “mise en forme” d'agents dans un cadre multi-agents. La mise en place d'une coordination entre agents peut s'avérer très difficile, du fait de la rareté des événements à récompenser (quand quatre prédateurs encerclent une proie pour la première fois, par hasard). Ce problème semble donc assez intuitivement justifier l'utilisation d'une "aide" pour guider l'apprentissage. Nos travaux vont présenter deux aspects différents : d'abord un progrès[32] en **complexité** de la tâche à accomplir (section 4.2), puis un progrès en **nombre** d'agents impliqués (section 4.3). Nous essaierons finalement de prendre du recul sur ces approches à travers une discussion en section 4.4.

4.2 Progrès en complexité

4.2.1 Principe

4.2.1.1 Dans les grandes lignes

Cherchant à concevoir des agents coopérants, et sans avoir eu vent des travaux sur le "shaping" sus-cités, nous avons naturellement commencé à mettre au point un apprentissage progressif passant par deux étapes. Nous ne nous intéressons pour l'instant qu'à la première (voir en section 4.3 pour la seconde).

La première étape suppose qu'on considère un nombre réduit (éventuellement minimal) d'agents, mais suffisant à l'accomplissement de la tâche collective. On va alors adopter une méthode comparable à celle d'Asada & al. vue en section 4.1.3, c'est-à-dire placer les agents d'abord dans des situations simples, puis dans des situations progressivement de plus en plus complexes.

Pour préciser notre cadre de travail, les agents n'ont qu'une observation partielle de leur environnement et apprennent de manière indépendante les uns des autres. On suppose la fonction de récompense assez bien définie pour amener les agents à coopérer (c'est une hypothèse forte, mais dont on a vu (section 3.5) qu'elle est incontournable si chaque agent doit avoir une fonction de récompense "interne").

4.2.1.2 De manière plus formelle

Pour décrire plus en détail la méthode, définissons quelques termes :

1. Un **essai** est une suite de n pas[33] commençant dans une *situation initiale* donnée. Un essai permet donc de mener une courte exploration depuis le point de départ défini.
2. Une succession de N essais sera appelée une **expérience**. Ce n'est que grâce à une telle répétition d'essais que les agents pourront accomplir une exploration utile du voisinage de la situation choisie, et donc réellement améliorer les comportements individuels.
3. En construisant une séquence d'expériences de difficultés progressives, on peut alors définir un **entraînement** pour aider à la conception de l'agent.

Le schéma opératoire est présenté à travers l'algorithme 4. Comme on peut le constater, une fois l'entraînement effectué, on laisse les agents continuer leur apprentissage jusqu'à ce qu'un critère d'arrêt choisi soit vérifié. Il peut même s'agir d'un apprentissage continuel si l'on suppose que l'environnement peut évoluer et nécessiter une adaptation des agents.

4.2.1.3 Pourrait-on automatiser la mise au point de l'entraînement ?

Comme expliqué dans [Asada et al, 1996], on peut automatiser *la sélection des situations* dans lesquelles placer les agents (même s'il est alors question d'un cadre mono-agent) à condition que

[32] Dans [Dutech *et al.*, 2001], il était sujet d'*apprentissage incrémental*, mais c'était avant de connaître une terminologie apparemment plus appropriée.

[33] Pendant un *pas* de temps, tous les agents agissent simultanément.

Algorithme 4 Apprentissage progressif en complexité

Note : la phase d'apprentissage des agents n'est pas explicite ici. Seules les entrées (perceptions) et sorties (actions) des agents apparaissent.

Entrées: Un système Σ constitué d'un ensemble $\mathcal{A}$ d'agents et d'un environnement.
Un entraînement $\mathcal{E}$.

1: **pour tout** expérience E de l'entraînement $\mathcal{E}$ **faire**
2: e = essai défini dans E par :
3: $\star$ s_0 une situation initiale dans laquelle placer le système
4: $\star$ n un nombre de pas à effectuer depuis s_0
5: N = nombre de répétitions de e à effectuer
6: **pour tout** essai $i \in [1..N]$ **faire**
7: Placer le système dans la situation s_0
8: **pour tout** pas $j \in [1..n]$ **faire**
9: **pour tout** agent $\alpha \in \mathcal{A}$ **faire**
10: Donner à α ses perceptions o_t^α
11: Demander à α sa décision a_t^α
12: **fin pour**
13: Simuler l'évolution du système : $\Sigma_t \rightarrow \Sigma_{t+1}$
14: $t \leftarrow t+1$
15: **fin pour**
16: **fin pour**
17: **fin pour**
18: **répéter**
19: [les lignes 9 à 14]
20: **jusqu'à** critère d'arrêt de l'apprentissage vérifié

Sorties: Les politiques des agents de $\mathcal{A}$.

l'on puisse établir un ordre (même grossier) sur ces situations. N'ayant pas fait d'hypothèses de ce type sur les systèmes que nous considérons, les situations initiales des différents essais doivent être définies manuellement.

On pourrait chercher aussi à automatiser *le choix du nombre n* de pas de temps alloué à chaque essai. Il faudrait pour cela évaluer en combien d'étapes les agents peuvent atteindre leur but (si l'on suppose le but "ponctuel") ou une zone dans laquelle le comportement est bien établi, de manière à ce que les explorations soient assez longues pour que l'apprentissage se fasse. Mais cela nécessiterait la connaissance d'un modèle du PDM sous-jacent (en supposant le contrôle centralisé), ce qui n'est pas toujours le cas : si l'on sait simuler le système, en déduire un modèle $\mathcal{T}$ manipulable n'est pas une tâche simple pour autant (de part la grande taille qu'il pourrait atteindre).

[Asada et al, 1996] propose aussi de changer d'expérience une fois le comportement suffisamment stabilisé (se passant ainsi de la définition du *nombre N* d'expériences à effectuer). Une raison pour ne pas l'avoir fait est que définir N à la main n'ajoute pas beaucoup de travail une fois les situations et nombres de pas déjà déterminés. Mais nous verrons une autre justification en section 4.2.2.2.

4.2.1.4 Autres remarques

Ce principe de faire suivre aux agents un entraînement n'est pas plus spécifique au cas mono- ou multi-agents. Il peut être utile dans les deux cas mais le sera évidemment d'autant plus dans des cadres mettant en jeu plusieurs agents devant collaborer.

Un autre point à noter est qu'accomplir un entraînement nécessite la capacité de placer le système dans les situations initiales particulières. Ainsi, de manière générale, il va falloir travailler sur des simulateurs, avec tous les problèmes de fiabilité que cela entraîne. C'est d'ailleurs parce que le passage en situation réelle change l'environnement des agents qu'il peut être intéressant de les laisser continuer à apprendre, ce qui leur permet de continuer à s'adapter.

4.2.2 Résultats

Cette approche progressive selon la difficulté de la tâche a été évaluée sur deux problèmes. Le premier, "fusion de blocs", est original, et permet d'utiliser un environnement dans lequel le nombre d'agents et autres objets impliqués et variable. Le second est un problème prédateurs-proie plus classique. Ces deux problèmes ont été choisis parce qu'ils nécessitent l'un comme l'autre une réelle coordination des agents.

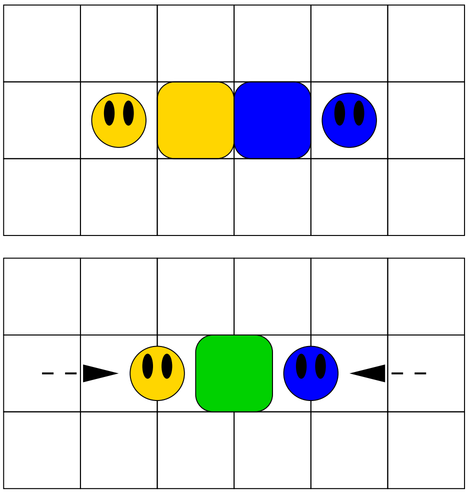
4.2.2.1 Fusion de blocs

Description du problème

La tâche choisie met en œuvre des agents de deux types, jaunes ou bleus, dans un monde pavé plan. Leur but est de pousser des cubes jaunes contre des cubes bleus[34], comme illustré sur la figure 4.5.

Quand deux agents coordonnent leurs mouvements pour atteindre ce but (pousser ensemble une paire de cubes), les deux cubes disparaissent temporairement pour réapparaître de manière aléatoire autre part sur le pavage. Simultanément, les agents responsables de cette "fusion" reçoivent alors une récompense positive. L'objectif des agents est ainsi de provoquer de telles fusions le plus souvent possible.

[34] La couleur des agents est ici sans lien avec celle des cubes.



Ici, agent et bloc de gauche sont en jaune, et bloc et agent de droite sont en bleu (mais les couleurs des blocs pourraient être échangées).

On a représenté en vert les blocs fusionnant avant de disparaître. Il est à noter que les **deux** agents doivent pousser les blocs pour que l'opération soit effectuée. Il ne suffit pas que l'un pousse et l'autre bloque.

FIG. 4.5 – Un exemple de fusion de blocs.

Description des agents

actions : En ce qui concerne les capacités des agents, ils ne disposent que de quatre actions possibles, correspondant à des déplacements d'une case vers le nord, l'ouest, le sud et l'est (un agent cherche toujours à se déplacer). Les agents peuvent pousser d'autres agents et d'autres cubes (même plusieurs), ce qui rend les conséquences de leurs actions stochastiques. En effet, quand plusieurs agents ont une influence sur le déplacement d'un bloc (par exemple), une seule de ces influences va avoir un réel effet. Ainsi, une part de l'incertitude est liée à l'ordre aléatoire de la résolution de ces contraintes.

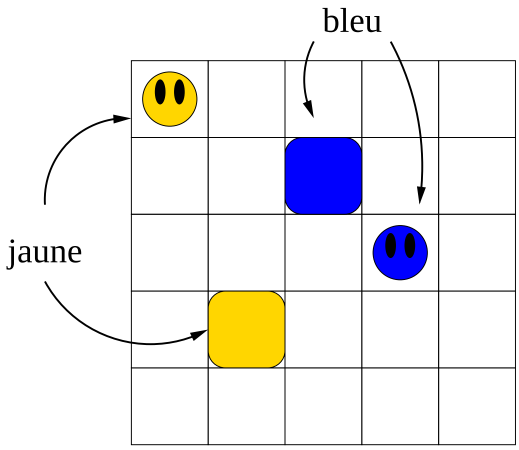
perceptions : Comme le montre la figure 4.6, les perceptions d'un agent sont constituées des informations suivantes :

- `dir(aco)` : La direction de l'agent de couleur opposée le plus proche parmi les quatre directions cardinales `(N-O-S-E)`.
- Pour le bloc jaune le plus proche :
  - `dir(bj)` : Sa direction `(N-NO-O-SO-S-SE-E-NE)`.
  - `proche(bj)` : Est-il sur l'une des huit cases entourant l'agent `(vrai|faux)` ?
- Pour le bloc bleu le plus proche :
  - `dir(bb)` : Sa direction `(N-NO-O-SO-S-SE-E-NE)`.
  - `proche(bj)` : Est-il sur l'une des huit cases entourant l'agent `(vrai|faux)` ?

En combinants ces perceptions, on arrive à un maximum de 1024 observations. Mais parmi celles-ci, un certain nombre correspondent à des combinaisons impossibles (on ne peut avoir un bloc jaune et un bloc bleu tous deux proches et à l'ouest). On peut ramener ce nombre à 256 en utilisant les symétries du problème (celui-ci est invariant par rotation, d'où une possible division par quatre). C'est à comparer avec les 15.249.024 états du problème centralisé et complètement observé dans un environnement de taille 8×8, d'autant que cet ensemble d'observations est *indépendant de la taille du monde.*

récompenses : Pour ce qui est de la fonction de récompense choisie, elle donne un renforcement positif (+5) à chacun des agents ayant été à l'origine d'une fusion de blocs, et aucun renforcement le reste du temps (0). On peut remarquer qu'il s'agit bien ici, comme souhaité, d'une récompense dépendant du point de vue local de l'agent.

Supposons que l'on considère l'agent jaune, l'agent bleu étant au nord (on peut toujours



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

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

FIG. 4.6 – Exemples de perceptions (deux agents dans un monde simple).

se ramener à ce cas). La récompense +5 sera donnée pour les transitions suivant le schéma suivant (à une permutation des blocs jaunes et bleus près) :

$$\begin{pmatrix} oa : \texttt{loin\_au\_nord} \\ bj : \texttt{près\_au\_nord} \\ bb : \texttt{loin\_au\_nord} \end{pmatrix} \xrightarrow[\texttt{aller au nord}]{} \begin{pmatrix} oa : \texttt{près\_au\_nord} \\ bj : \quad * \\ bb : \quad * \end{pmatrix} \tag{4.2}$$

On peut noter ici que la récompense définie ne pénalisera jamais un agent qui empêche ses congénères de travailler : chacun pense principalement à son propre gain, et non à celui du groupe.

Remarque : Pour que le problème posé aux agents puisse être résolu et n'amène pas à des situations bloquées, il a fallut interdire que les blocs puissent aller sur une des cases bordant la grille (laquelle n'est pas torique). Sans cela, les blocs ne pourraient plus être replacés vers le centre de l'environnement, et risqueraient même de se retrouver immobilisés dans un des quatre coins.

Pour terminer, précisons que les agents apprennent à l'aide de l'*algorithme de montée de gradient (en-ligne)* décrit en section 2.3.3.2.

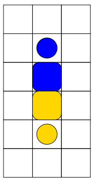
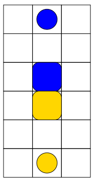
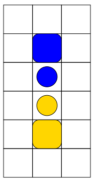
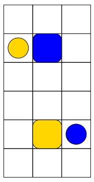
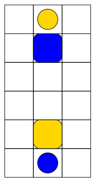
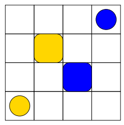
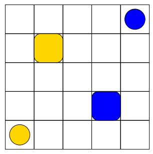
Méthodologie

Si le problème de fusion de blocs présenté peut mettre en jeu un nombre a priori variable de blocs et d'agents, l'apprentissage progressif en complexité qui nous intéresse ne sera utilisé que sur un dispositif réduit au minimum utile : un agent et un bloc de chaque couleur.

La table 4.1 montre une séquence d'expériences utilisée pour aider les agents dans leur entraînement. La première *situation initiale*, dans un monde 6×3, n'a besoin que d'un mouvement pour que le but soit atteint, chaque agent poussant dans la direction des blocs. Toutefois, ils ont en tout 6 pas de temps pour accomplir cette tâche, ce qui leur permet d'explorer différents mouvements et leurs conséquences. Un tel essai est ici répété 750 fois.

Note : pour ne pas favoriser l'un ou l'autre agent, on a choisi d'utiliser des situations toujours symétriques. Mais cela n'empêche pas les spécialisations.

TAB. 4.1 – La séquence d'expériences utilisée pour l'apprentissage progressif.
Note : On peut voir ici qu'il n'est pas évident de déterminer si les tâches sont réellement ordonnées.

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

Pour pouvoir juger de l'efficacité de l'approche adoptée, un apprentissage standard est comparé à l'apprentissage progressif étudié. Dans les deux cas, on compte le nombre de fusions de blocs effectuées pendant 1000 pas de temps pour évaluer la qualité des politiques adoptées par les agents, ceux-ci étant placés dans un environnement de taille 8×8 une fois l'éventuel entraînement terminé. Nous allons maintenant observer et analyser les résultats obtenus.

Résultats

On peut voir sur la figure 4.7 les évolutions des mesures de performance du groupe de deux agents avec et sans entraîneur (moyenne sur 10 simulations indépendantes). La courbe montrant l'efficacité de l'apprentissage après une période d'entraînement ne commence qu'une fois passés les 12000 pas de temps (on parle aussi de "coups") que dure le script du tableau 4.1.

Dans les deux cas, les agents tendent vers des comportements leur permettant d'effectuer près de 90 fusions sur 1000 pas de temps (c'est-à-dire une fusion tous les 11 pas environ). Par contre, une fois le délai de l'entraînement passé, la convergence vers de très bonnes performances se fait notablement plus vite que par un apprentissage "standard". Si les agents n'ont pas tout appris via le script qui les guide pendant les premiers temps, au moins en ont-ils bénéficié pour efficacement "mettre en forme"[35] leurs politiques.

La séquence d'expériences présentée ici ne se veut pas optimale. Elle a été établie sans chercher à affiner le résultat obtenu, ce qui aurait été un travail long et sans intérêt particulier. Néanmoins, il faut noter que la durée des différentes phases doit être en adéquation avec la vitesse d'évolution des différents paramètres de l'apprentissage (surtout le paramètre α en l'occurrence, paramètre présenté en section 2.2.2.3). Certains essais effectués laissaient les agents ancrer certains apprentissages trop profondément (décisions à prendre dans le cas d'une certaine observation o par exemple), alors que d'autres expériences ultérieures devaient leur faire reconsidérer ces connaissances (cette même observation o appelant à adopter un comportement tout autre).

[35]"to shape" peut se traduire par "mettre en forme".

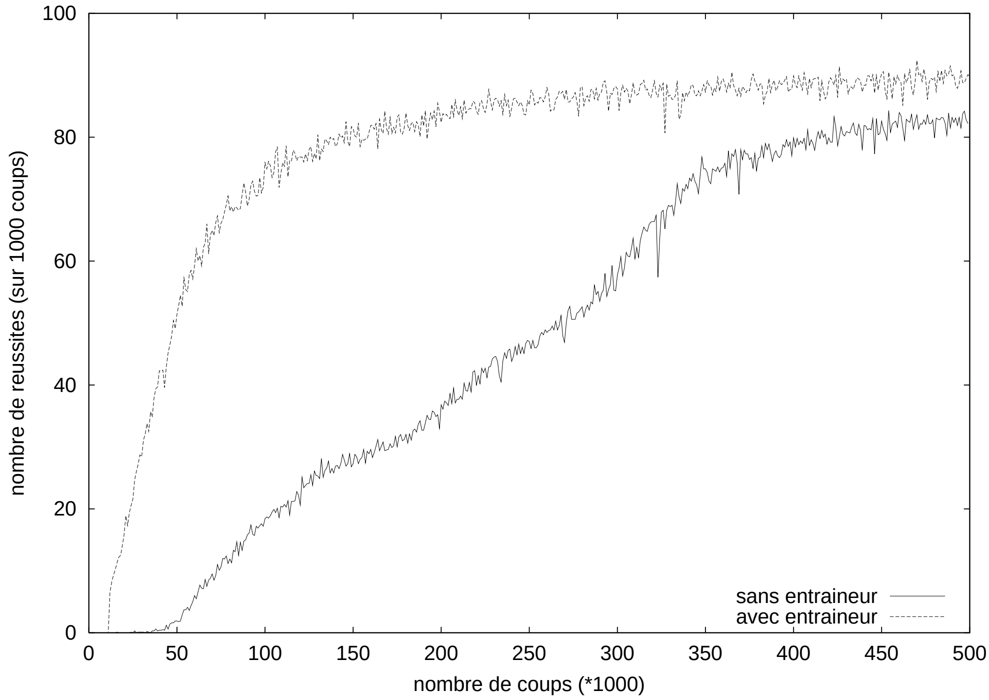


FIG. 4.7 – 2 agents, 2 blocs : apprentissage standard vs. apprentissage progressif

4.2.2.2 Prédateurs-Proie

Description du problème

Dans ce second exemple, le système multi-agents considéré doit aboutir à une coordination impliquant quatre agents coopérant à la fois (sans compter la victime). En effet, le problème posé, dû à [Benda *et al.*, 1986] (et parfois appelé "poursuite"), est d'encercler une proie pour l'empêcher de bouger, les prédateurs étant donc au nombre de quatre.

On se place encore une fois dans un environnement plan quadrillé, mais désormais torique (un agent sortant d'un côté du terrain réapparaît du côté opposé), de façon à ce que l'on ne puisse pas coincer la proie contre un mur, ce qui changerait considérablement le problème.

La figure 4.8 montre d'une part une situation de jeu quelconque dans un monde de dimensions[36] 7×7, et d'autre part une situation de proie attrapée dans le même environnement.

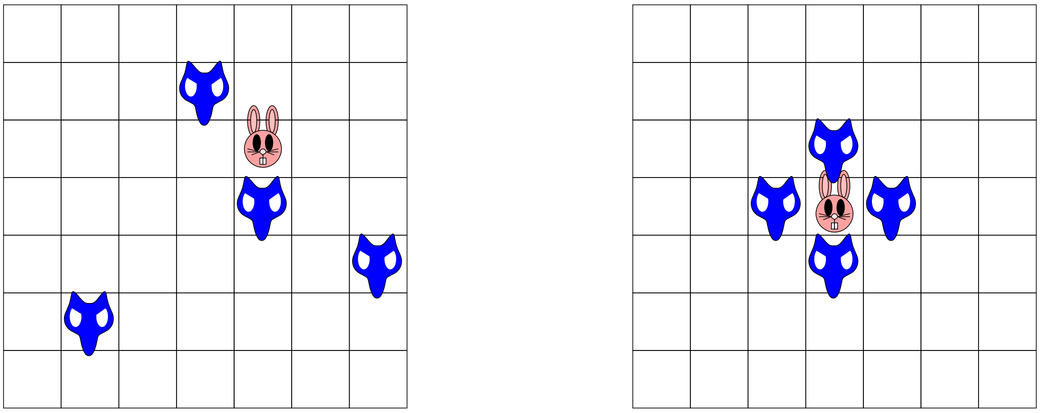
Un point à noter est que l'agent "proie" ne cherche pas particulièrement à fuir. Il est suffisant qu'il adopte un comportement aléatoire pour que la tâche soit difficile.

Description des agents

actions : Cette fois-ci, en plus des quatre déplacements élémentaires, l'ensemble des actions disponibles contient l'action "ne pas bouger".

L'incertitude quant au résultat de ces actions est due ici non seulement à la part de hasard dans les résolutions de contraintes, mais aussi à un bruit ajouté artificiellement. Si ce bruit est effectivement ajouté, le déplacement choisi est 8 fois sur 10 obtenu, les déplacements

[36]Les dimensions de ce monde *torique* sont toujours impaires, pour qu'il n'y ait jamais d'ambiguïté sur la direction dans laquelle est vue une case aux antipodes d'une autre.



a- situation quelconque b- situation recherchée

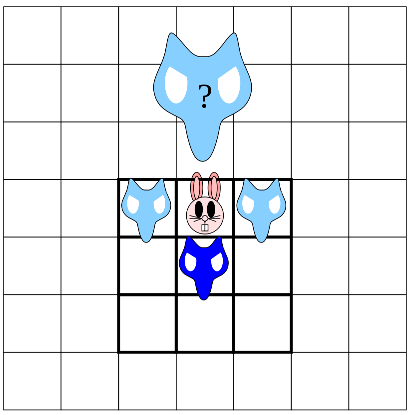
FIG. 4.8 – Deux exemples de situations possibles du problème prédateurs-proie.

“voisins” étant obtenus de manière équiprobable le reste du temps (voir figure 2.3 en section 2.2.1.1).

perceptions : Les perceptions d'un prédateur lui donnent les mêmes informations sur ses trois congénères et sur la proie :
- `dir(.)` : Sa direction `(N-NO-O-SO-S-SE-E-NE)`.
- `proche(.)` : Est-il sur l'une des huit cases entourant l'agent `(vrai|faux)` ?

On a ainsi 16 cas possibles pour chacun des quatre agents perçus, donc en première estimation $16^4 = 65536$ observations possibles. En utilisant les symétries du problème et le fait que les trois prédateurs ne sont pas distingués, on peut se ramener à 1324 observations, dont quelques unes sont impossibles (comme dans le problème de fusion de blocs).

récompenses : Ici encore, on va définir une fonction de récompense individuelle ne dépendant que des perceptions locales de chaque agent. Un prédateur ne reçoit un signal de renforcement non nul (fixé en l'occurrence à 25) que quand l'observation peut correspondre à la prise d'une proie, comme l'illustre la figure 4.9.



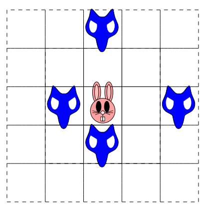
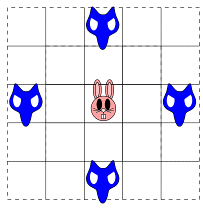
Pour le prédateur ici au sud, les comportements à renforcer sont ceux qui amènent à une situation avec la proie entre lui et deux autres prédateurs, le quatrième étant de l'autre côté de la proie (mais on ne peut être sûr qu'il y ait encerclement).

FIG. 4.9 – Ce que perçoit un prédateur quand il reçoit un signal de renforcement.

Méthodologie

La table 4.2 montre une séquence d'expériences utilisées pour l'entraînement des agents prédateurs. A chaque fois, l'entraînement est effectué dans un monde 7×7 (images ici réduites). Pour conserver l'idée d'équité dans l'expérience des agents, et comme cette fois-ci l'une des situations n'est pas symétrique, les prédateurs sont à chaque fois placés dans l'un ou l'autre rôle.

TAB. 4.2 – La séquence d'expériences utilisée pour l'apprentissage progressif.

Situation initiale		
n (pas)	4	8
N (essais)	500	500

La méthodologie pour évaluer l'efficacité de cette approche est la même que précédemment. On va comparer un apprentissage standard à l'apprentissage progressif étudié. De manière similaire au problème de fusion de blocs, on va ici compter le nombre de proies attrapées pendant 1000 pas de temps, les agents étant placés dans un environnement de taille 7×7 une fois l'éventuel entraînement terminé.

Résultats

On peut voir sur la figure 4.10 les évolutions des mesures de performance du groupe des quatre prédateurs avec et sans entraîneur (moyenne sur 10 simulations indépendantes). La courbe montrant l'efficacité de l'apprentissage après une période d'entraînement ne commence qu'une fois passés les 1000 pas de temps que dure le script du tableau 4.2.

Toutefois, notre approche progressive semble ne pas bien fonctionner. Ce phénomène est lié aux observations partielles, qui font qu'une partie du comportement appris pendant la phase d'entraînement doit être révisée après coup (les observations pendant l'entraînement n'étant pas représentatives des états sous-jacents rencontrés à l'exécution). Cela devient plus une perte de temps qu'un gain. La figure 4.11 illustre ce problème sur un exemple simple.

Après le problème de fusion de blocs, ce second exemple montre que, dans un cadre partiellement observable, il faut aussi se soucier de trouver un entraînement qui n'implique pas de désapprendre par la suite.

4.3 Progrès en nombre

4.3.1 Principe

4.3.1.1 Dans les grandes lignes

Comme nous l'avons annoncé, nous n'avons pas limité notre approche progressive de l'apprentissage dans un système multi-agents à l'évaluation d'une méthode d'entraînement comparable à celle de [Asada et al, 1996], mais avons aussi proposé une deuxième étape dans laquelle le progrès se fait par la croissance du nombre d'agents (et autres objets) présents dans l'environnement (ainsi qu'à l'éventuelle changement de taille du dit environnement).

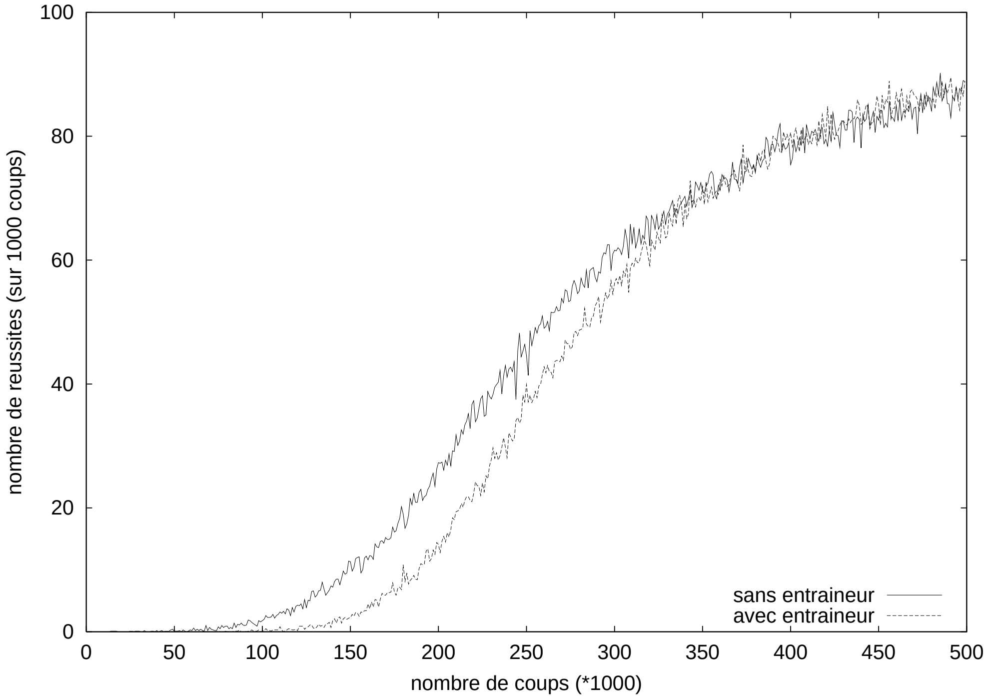
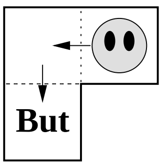


FIG. 4.10 – 4 prédateurs-1 proie : apprentissage standard vs. apprentissage progressif.



Supposons un agent parcourant un labyrinthe en aveugle (une seule perception pour toute case). S'il est d'abord entraîné depuis la case juste au dessus du but, il va ensuite lui falloir désapprendre pour adopter une politique plus efficace dans la situation représentée ci-contre.

FIG. 4.11 – Un cas problématique : un simple labyrinthe parcouru en aveugle.

Si le comportement à adopter n'est pas strictement le même quand le nombre d'objets présents dans l'environnement augmente, l'idée suivie ici se fonde sur le fait qu'un comportement déjà appris dans une situation peut être un bon point de départ pour d'autres situations comparables.

Contrairement à ce qui a été fait en section 4.2, aucun algorithme précis ne sera fourni ici. La plupart des essais menés pour évaluer cette méthode partiront de comportements appris dans un cadre à 2 agents et 2 blocs pour les réutiliser et les adapter dans des environnements "enrichis".

4.3.1.2 Quelques remarques

mono ou multi ? - Ici encore, on adopte une méthode qui n'est a priori pas limitée à des agents vivant en groupe. Si on se restreint au nombre d'objets ou à la taille de l'environnement, le principe peut déjà s'appliquer à nombre de problèmes mono-agents.

"scalabilité" - Un point que soulève ce "progrès en nombre" est la capacité des agents à s'adapter à des environnements de dimensions variables (taille comme nombre d'objets contenus). On parlera alors de *capacité de mise à l'échelle* ou encore de *"scalabilité"* ou d'*agent "scalable"*. Il s'agit là d'affreux anglicismes mais, si la traduction du terme anglais est possible, elle s'avère très lourde à employer[37].

perceptions - Cette "scalabilité" requiert l'adoption d'une architecture adéquat par les agents. Un agent peut être conçu pour une forme d'environnement donnée, alors qu'ici il va être nécessaire de pouvoir réutiliser le même comportement dans les différentes formes que peut prendre un environnement. La réponse à ce problème peut être très dépendante du cas considéré, des approximations que l'on accepte de faire... Nous verrons simplement le choix qui a été suivi dans nos expérimentations (ne considérer que les objets les plus proches) et le discuterons.

4.3.2 Résultats : Fusion de blocs

Description du problème

Le problème est le même que dans la section 4.2.2.1, à la différence que pourront être considérés plus d'un agent ou bloc de chaque couleur (mais il en faut toujours au moins un de chaque).

Description des agents

actions : `[inchangées]`

perceptions : La description déjà faite des perceptions d'un agent restent valables. Il faut simplement remarquer qu'elles permettent, en s'intéressant aux objets les plus proches, d'être compatibles avec des environnements de dimensions variables. Ce critère de sélection des objets sur lesquels se porte l'attention d'un agent est arbitraire (règle qu'on pourrait qualifier de "réactive"). Des solutions plus élaborées pourraient être employées.

On peut noter aussi que le point de vue subjectif adopté (voir section 1.1.5.2) permet d'opérer dans des environnements de tailles diverses, contrairement à ce que permettrait l'énumération complète d'états dans un processus de décision markovien classique.

récompenses : `[inchangées]`

[37] J'espère ne pas trop choquer les lecteurs de ces lignes.

Méthodologie

Comme précédemment, l'apprentissage avec progression va être ici comparé à l'apprentissage sans progression (standard). On va toutefois commencer par regarder l'efficacité des politiques apprises dans le cas des expérimentations précédentes (à 2 agents et 2 blocs) quand elles sont réutilisées dans des mondes plus peuplés.

Notations : Pour des raisons pratiques, on notera $\mathbf{n}a\mathbf{m}c$ une situation à n agents (autant de chaque couleurs) et m blocs (autant de chaque couleur aussi, la lettre c signifiant “cubes”). En ne considérant que les situations de cette forme, on restreint certes les analyses possibles, mais on verra qu'on peut déjà faire une étude intéressante.

Que ce soit dans le cas d'une simple réutilisation ou pour procéder à un apprentissage progressif, ce sont des politiques apprises dans la situation $2a2c$ qui seront employées. On a alors seulement les politiques d'un agent jaune et d'un agent bleu : il suffit de les reproduire dans les $n/2$ agents jaunes d'une part et les $n/2$ agents bleus d'autre part (processus illustré par la figure 4.12).

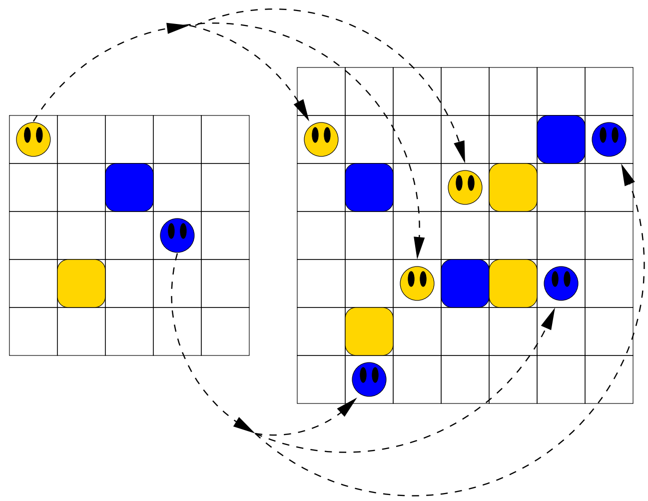


FIG. 4.12 – Réplication des comportements de deux agents dans n agents.

Résultats : simple réutilisation

On commence donc par utiliser des agents dont le comportement fixé a été appris dans le cas $2a2c$, ce qui n'est pas supposé être particulièrement efficace avec plus de cubes. Néanmoins, cela leur donne d'assez bons comportements pour obtenir quelques bons résultats.

Différents tests ont été menés avec 2, 4, 6, 8 ou 10 cubes et 2, 4, 8, 16 ou 20 agents (toujours dans un monde de taille 10×10). Nous avons utilisé des séries de 1000 pas de temps pour évaluer les performances des agents, chacune commençant dans des configurations aléatoires. Comme des blocages peuvent apparaître (l'ensemble des agents oscillant autour d'une situation donnée sans trouver de solution), 100 séries sont effectuées à chaque fois pour en déduire une efficacité moyenne.

La table 4.3 donne l'efficacité moyenne dans chacun des 25 cas (l'efficacité étant le nombre de fusions effectuées en 1000 coups). Les résultats montrent que, pour un nombre donné de cubes, il semble y avoir un nombre optimal d'agents, comme indiqué par les mesures marquées en gras dans la table. On remarquera dès à présent que, opérant dans un environnement plus grand qu'en section 4.2.2.1, le cas $2a2c$ apparaît ici moins productif qu'avant (efficacité moyenne de $40,4$ au lieu de près de 90).

Un nombre croissant d'agents améliore les résultats jusqu'à ce qu'ils se gênent les uns les autres et amènent plus de problèmes de coordination qu'ils n'en résolvent. Avec un nombre croissant de blocs, les résultats s'améliorent seulement jusqu'à un nombre optimal au-delà duquel on peut voir les agents hésiter sur le choix des cubes avec lesquels travailler et ainsi tomber facilement dans des séquences de mouvements oscillants.

cubes	agents				
↓	2	4	8	16	20
2	**40,4**	30,0	20,0	12,7	11,0
4	7,6	17,1	**17,5**	13,9	12,9
6	3,4	11,2	14,7	15,7	**16,5**
8	1,9	8,6	13,5	15,9	**18,0**
10	1,6	6,7	11,0	17,7	**20,6**

TAB. 4.3 – Efficacités moyennes (nombre de fusions pour 1000 coups)

cubes	agents				
↓	2	4	8	16	20
2	4,0	3,1	2,9	2,8	2,4
4	7,0	6,6	4,3	3,0	3,3
6	3,4	5,4	4,5	3,4	3,4
8	2,3	4,6	4,5	4,1	3,8
10	1,9	3,6	4,5	4,1	4,7

TAB. 4.4 – Ecarts types

La table 4.4, pour sa part, fournit les écart-types dans les mêmes situations (pour les mêmes séries d'expérimentations). Elle permet de voir que, quand le nombre croissant de cubes surcharge les agents (par exemple pour 2 ou 4 agents), la valeur de l'écart-type est du même ordre que l'efficacité moyenne. Il s'agit de situations dans lesquelles les agents sont souvent bloqués dans des mouvements cycliques.

Remarque : Une autre série d'essais a été effectuée dans les mêmes conditions mais avec des agents dénués de politiques (ou plus précisément dotés de politiques faisant des choix d'actions équiprobables). Cela a permis de vérifier que des comportements aléatoires ont toujours de mauvais résultats, quelque soit le nombre d'agents. Il ne suffit donc pas d'avoir un niveau d'agitation suffisant dans un groupe d'agents (que l'on pourrait qualifier de "browniens") pour effectuer la tâche souhaitée.

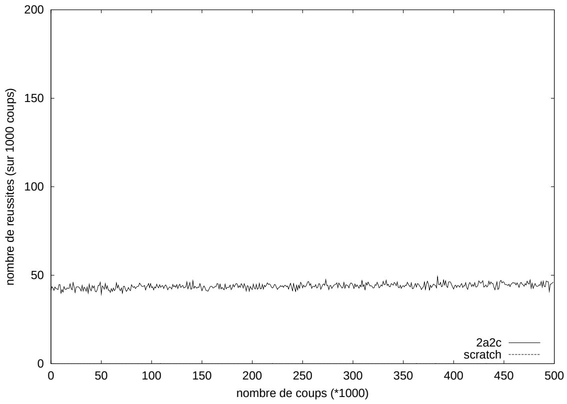
Résultats : apprentissage progressif

Si, comme nous venons de le voir, des comportements appris pour une situation à 2 agents et 2 cubes peuvent être réutilisés dans des environnements plus encombrés, il peut être intéressant de s'en servir de base pour adapter les agents à cette nouvelle situation en prolongeant l'apprentissage, ce qui constitue notre proposition d'apprentissage progressif en nombre.

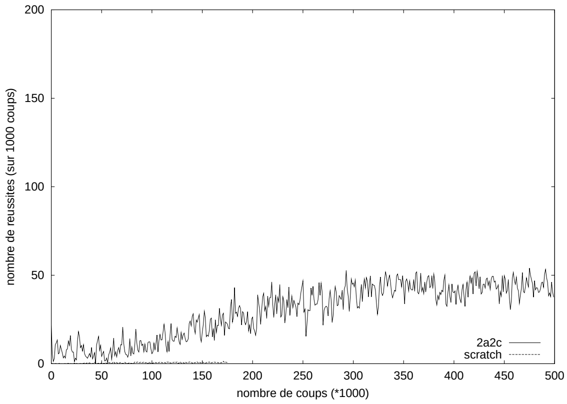
Pour évaluer cette méthode, des apprentissages d'agents dans différentes situations ont été comparés selon qu'ils étaient accomplis à partir d'agents néophytes ou à partir d'agents dotés d'une politique $2a2c$. Les figures 4.13 a, b, c et d montrent les résultats dans quatre situations distinctes.

On peut noter que dans les deux premiers cas, les agents néophytes ont d'importantes difficultés à apprendre quoi que ce soit. Il ne faut pas oublier qu'on est ici dans un environnement plus grand que dans la section 4.3 (10×10 au lieu de 8×8), ce qui rend les actions récompensées bien plus rares et par conséquent l'apprentissage plus difficile. D'ailleurs, de manière générale, la vitesse d'apprentissage augmente avec le rapport $n_{objets}/taille_{\text{monde pavé}}$ (les pentes sont de plus en plus fortes si l'on suit les quatre figures).

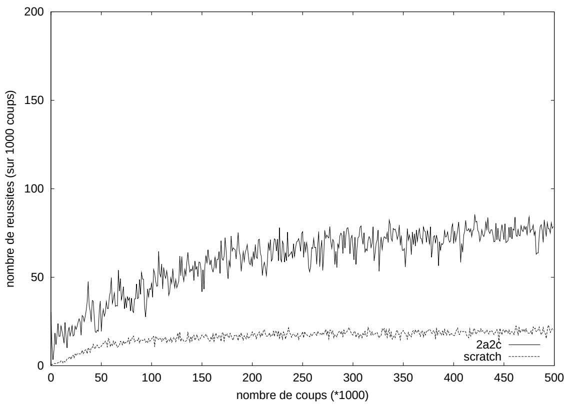
Pour revenir au problème de départ de cette étude, ces essais confirment clairement que des agents ayant déjà l'expérience de la politique à adopter dans un problème du même type ont



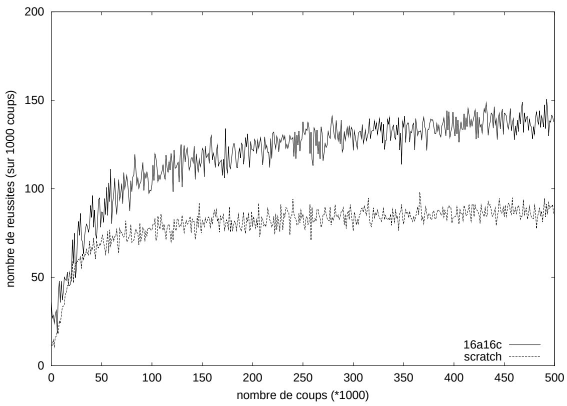
a- $2a2c$



b- $4a4c$



c- $8a8c$



d- $16a16c$

FIG. 4.13 – Comparaison entre apprentissage standard (à partir de zéro) et apprentissage progressif (à partir de $2a2c$) pour différents environnements.

une progression beaucoup plus rapide que s'ils étaient de réels débutants. Dans le cas a, on est même dès le départ à un niveau d'efficacité optimal (ce qui n'a rien d'étonnant puisque ce sont quasiment les conditions de l'apprentissage initial).

Outre la vitesse d'évolution du groupe, l'efficacité maximale atteinte par les agents via la méthode progressive est de loin meilleure que les performances obtenues par un apprentissage en partant de zéro. Comme on pouvait le prévoir, des agents utilisant des comportements $2a2c$ viennent avec des connaissances qui leur permettent de trouver un meilleur optimum local[38].

Dernière remarque : Il faut garder à l'esprit que les perceptions utilisées font qu'un agent n'a pas de bonne perception de l'ensemble du groupe et de tous les cubes environnants. Ainsi, les grands groupes d'agents sont difficiles à analyser, puisque leurs comportements tiennent pour partie d'une agitation hasardeuse et pour partie d'un apprentissage (dont on a montré qu'il est réellement utile).

4.4 Discussion

4.4.1 Résultats obtenus

Autant l'apprentissage progressif en complexité (section 4.2) qu'en nombre (section 4.3) se sont montrés efficaces pour accélérer l'apprentissage, voire pour dépasser des optima locaux (qui peuvent exister puisqu'on est dans un cadre non-markovien). Une limitation assez clairement apparue est le risque de devoir "désapprendre" en partie le comportement obtenu par l'entraînement, problème lié à l'observabilité partielle de nos situations.

En outre, on peut se demander si un guidage ainsi conçu ne risquerait pas aussi, à l'inverse, de faire tomber dans certains optima locaux. Il serait par exemple facile d'indiquer un chemin trop long à un robot qui doit trouver la sortie d'un labyrinthe. S'il ne remet pas suffisament en cause ses apprentissages (par l'exploration), il ne sortira pas de son mauvais acquis.

4.4.2 Centralisation

Si on se réfère à la "catégorisation" faite dans la section 1.3.2.3 du premier chapitre de ce mémoire, on s'est clairement intéressé à un système multi-agents décentralisé à l'exécution (chaque agent est doté d'un comportement autonome). Mais la phase de conception passe par un entraîneur qui amène une centralisation, autant par le guidage que la simulation qu'il oblige d'utiliser.

Si la conception est déjà pour partie centralisée, une idée pertinente serait de ne concevoir qu'un seul "cerveau" commun à tous les agents, ce qui réduirait notablemment les temps d'apprentissage. Dans le cas proie-prédateur par exemple, à chaque pas de temps c'est de l'expérience des quatre prédateurs que bénéficiera leur cerveau unique.

Mais il faut voir que cela interdit des différenciations de rôles entre les agents, et qu'en simulation le temps n'est pas si critique. Cela pourrait toutefois être un moyen intéressant d'accélérer encore l'apprentissage, dans les cas où l'on ne craint pas de dégradations du comportement de groupe liées à la non-spécialisation des agents.

Note : Cette spécialisation est par exemple indispensable dans un problème où deux agents apprennent à trier *ensemble* une liste de trois chiffres (voir [Buffet, 2000]).

[38] Le fait de contourner les blocs dans certaines situations peut s'avérer difficile à apprendre dans un environnement bruité tel que $16a16c$, donc en apporter la connaissance de $2a2c$ peut être intéressant.

4.4.3 Scalabilité (capacité de mise à l'échelle)

La capacité de mise à l'échelle des agents, ou scalabilité, (le fait qu'ils puissent s'adapter à des environnements variés en taille comme en nombre d'objets l'occupant) requiert des perceptions simplement partielles, ainsi qu'une incapacité à distinguer les autres agents (ce qui empêche de "spécialiser" ses réactions par rapport à l'agent perçu).

Nous avons fait ici le choix de perceptions "orientées objets" et par voie de conséquence il a fallu définir un mécanisme de sélection des objets à considérer. Dans ce cas précis, on a arbitrairement opté pour ne garder que les perceptions correspondant aux objets les plus proches. Mais dans des cas plus classiques, tel qu'un robot doté de sonars par exemple, un tel problème de choix des perceptions à utiliser ne serait pas apparu.

Le problème de la conception d'agents "scalables" reviendra dans la troisième partie de ce mémoire puisque les travaux qui y sont présentés sont plus particulièrement guidés par cette possibilité de gérer un environnement complexe.

4.4.4 Automatisation

Le problème de l'automatisation de l'une ou l'autre phase de la méthode d'apprentissage progressif présentée a déjà été discuté en section 4.2.1.3. Il y a de quoi travailler sur cet aspect, mais l'intervention du concepteur semble devoir toujours être présente à un moment ou à un autre. Ne serait-ce que pour régler les paramètres de l'algorithme d'apprentissage utilisé, il y a toujours un niveau auquel il faut travailler manuellement (comme dans tout système d'optimisation).

4.4.5 Dernière réflexion

La séparation qui a été faite entre la progression en complexité et la progression en nombre peut d'une certaine façon paraître quelque peut artificielle. En effet, on pourrait plus simplement identifier ici 2 dimensions différentes de l'espace des problèmes auxquels on s'est intéressé ici. Là où [Asada et al, 1996] distinguait des sous-états tels que la *taille* et l'*angle* du but, c'est selon les deux dimensions de la *distance à la fusion de deux blocs* et du *nombre d'objets dans l'environnement* que peuvent être faites des progressions dans le cas de la fusion de blocs).

Conclusion

Cette deuxième partie a permis d'aborder la question de la complémentarité entre systèmes multi-agents et apprentissage par renforcement sous l'angle de l'utilisation de l'A/R pour concevoir des agents coopérants au sein d'un groupe.

Travail accompli

On a pu appronfondir à travers le chapitre 3 quelques problèmes soulevés dans ce cadre :

- Pour s'assurer de la collaboration des agents dans le sens souhaité, il faudrait savoir définir des fonctions de récompense individuelles qui reflètent l'objectif fixé au groupe. On n'en est hélas pour l'instant pas capable.
- Chaque agent ayant son propre point de vue (et entre autres des perceptions partielles), il leur est difficile de s'accorder sur des comportements joints à adopter. Le groupe tombe ainsi facilement dans des comportements sous-optimaux.
- Modéliser les autres permet souvent une meilleure adaptation d'un agent à ses congénères.
- Un problème de type DEC-POMDP est, de manière générale, d'une grande complexité temporelle.

Ces grandes difficultés nous ont amené à proposer une méthode pratique pour aider à la conception de systèmes multi-agents, tout en utilisant un algorithme de recherche directe de politique (puisque l'on ne peut raisonnablement considérer la situation des agents comme markovienne). Nous nous sommes ainsi dirigés vers les approches d'apprentissage progressif (*shaping*) dans le chapitre 4. Après une présentation du domaine en section 4.1, nous avons présenté et évalué une étape d'apprentissage progressif en complexité (du problème) (section 4.2), ainsi qu'une autre étape d'apprentissage progressif en nombre (d'objets et d'agents présents) (section 4.3).

Comme expliqué dans la discussion de la section 4.4, les deux étapes ne sont pas si distinctes l'une de l'autre que l'on pourrait le croire. Quoi qu'il en soit, elles s'avèrent assez efficaces (au problème de "désapprentissage" près) et feraient presque oublier qu'on a dû passer outre la centralisation de la phase de conception pour pouvoir ainsi améliorer l'apprentissage du groupe.

Problématique soulevée

Une notion importante qui est apparue est celle de "scalabilité", méchant anglicisme utilisé pour traduire le besoin qu'un agent soit *capable de se mettre à l'échelle* quand le nombre d'objets l'environnant ou les dimensions du problème auquel il est confronté changent, comme c'est souvent le cas dans un cadre multi-agents. Cette idée a en partie guidé les travaux présentés dans la troisième partie de ce mémoire à laquelle nous allons maintenant passer.

Troisième partie

Combinaison de comportements

Introduction

> « Les moyens peuvent être comparés à une graine et la fin à un arbre ; et il existe le même rapport intangible entre les moyens et la fin qu'entre la graine et l'arbre. »
>
> *Gandhi*

A l'inverse de la partie précédente, c'est à des approches multi-agents pour la conception d'un agent apprenant par renforcement que nous allons nous intéresser dans cette troisième et dernière partie. On pourra aussi parler plus simplement d'approches par décomposition.

Le chapitre 5 fera un aperçu assez rapide de l'existant dans ce domaine, lequel recouvre une grande variété de travaux. Une des difficultés que l'on rencontrera sera d'organiser ces travaux de manière à avoir une classification claire. Mais nous y reviendrons en temps utile.

Notre travail va en fait chercher à répondre à un problème précis apparu dans le chapitre 4 : comment réaliser un agent "scalable" (c'est-à-dire pouvant faire face à des situations plus complexes qu'à son habitude, mais de même nature) ? La question qui nous préoccupe est plus précisément de concevoir un agent capable de s'adapter à des environnements de dimensions variables, ces dimensions concernant entre autres le nombre d'objets présents. Cette question a été d'abord soulevée parce que nos agents (dans le problème de fusion de blocs vu en section 4.2.2.1) n'étaient pas en mesure de percevoir, et donc de gérer, des objets en nombre variable. Nous étions ainsi contraints d'utiliser une heuristique simpliste : choisir les objets les plus proches, pour ne sélectionner qu'un nombre fixe d'objets à prendre en considération dans les perceptions.

L'apport que nous proposons ici se base sur une situation particulière. On va faire l'hypothèse d'un agent confronté à une tâche complexe, fruit d'une accumulation de tâches simples. On verra donc dans le chapitre 6 comment on peut combiner des politiques dédiées à différentes sous-tâches de l'agent. Puis, les chapitres 7 et 8 montreront comment on peut efficacement apprendre un comportement plus complexe via cette méthode de combinaison et, de là, comment l'agent "combinant" peut acquérir une plus grande autonomie en trouvant de lui même les tâches simples qui vont servir de briques pour recomposer un comportement complexe.

Chapitre 5

SMA pour PDM

Sommaire

5.1 Introduction

Dans la deuxième partie de ce mémoire, l'apprentissage par renforcement a été vu comme un outil pour la conception de systèmes multi-agents. Dans cette troisième partie nous allons, de manière symétrique, nous intéresser à l'utilisation de la "méthodologie SMA" pour concevoir un système apprenant par renforcement (ce système pouvant lui aussi revendiquer l'appellation d'"agent").

Evoquons pour commencer le cas du Q-learning. Celui-ci peut en effet être présenté comme un algorithme mettant en jeu un *groupe* d'entités autonomes, chacune responsable de la prise de décision (du groupe) dans une situation donnée. Pourtant, on considère généralement un agent qui fonctionne selon le principe du Q-learning comme constituant une *unique* entité apprenante.

Dans le même ordre d'idée, [Meuleau et Dorigo, 2002] comme [Birattari *et al.*, 2002] ont étudié des phénomènes émergents au sein de groupes de fourmis (voir la présentation faite en section 1.2.2), c'est-à-dire des phénomènes dans lesquels la solution d'un problème est élaborée par le travail d'un groupe. De là, ils ont mis en évidence le lien qui peut être fait entre de tels phénomènes émergents et des algorithmes d'optimisation (appliquables à l'apprentissage par renforcement).

Mais, outre ces deux idées, on peut chercher à ré-organiser explicitement l'architecture interne d'un agent apprenant sous une forme multi-agents. Nous allons ainsi voir dans la section qui suit une vaste classe d'approches des PDM (puisque l'hypothèse de Markov est souvent faite), laquelle s'appuie sur l'idée de les décomposer d'une manière ou d'une autre. Après un aperçu de cette classe en section 5.2, nous allons présenter un domaine connexe en section 5.3, celui de la sélection d'action, et pourrons ainsi nous restreindre en section 5.4 aux approches par décomposition qui entrent dans ce cadre.

5.2 Décomposition de PDM

5.2.1 Principe

Les méthodes d'apprentissage par renforcement souffrent souvent d'être appliquées dans des espaces trop grands : soit on ne peut plus stocker les informations nécessaires, soit il devient irréaliste d'obtenir les expériences suffisantes pour un bon apprentissage, soit (à supposer qu'on travaille avec un modèle) les calculs sont simplement trop coûteux.

Devant de telles difficultés, une démarche possible est de "diviser pour régner", c'est-à-dire en l'occurrence de décomposer le problème d'apprentissage en morceaux plus petits et donc plus faciles à traîter.

5.2.2 Quelles sont ces approches ?

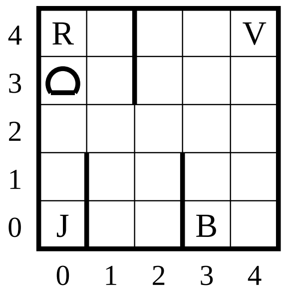
Une étude approfondie des approches par décomposition de l'apprentissage par renforcement serait trop laborieuse et encombrante pour être présentée ici. On se contentera donc de regarder assez brièvement les formes qu'elles peuvent prendre, sachant que la décomposition peut être menée selon différentes dimensions du problème, et de citer des travaux existant dans ce domaine.

On peut distinguer deux principales dimensions employées dans la décomposition d'un PDM : la tâche globale de l'agent est soit vue comme un ensemble de tâches **séquentielles**, soit comme un ensemble de tâches **concurrentes** ([Meuleau *et al.*, 1998] fait aussi cette distinction).

Des formes **séquentielles** simples sont les méthodes cherchant à scinder une tâche complexe en sous-tâches successives, les sous-PDM associés devant être exécutés l'un après l'autre (à chacun étant associée une condition de terminaison). Ce genre d'approches est souvent aussi présenté comme une structuration de politiques en sous-politiques, d'actions en sous-actions ou, dans l'autre sens, d'actions en macro-actions.

Il s'agit aussi de considérer le systèmes à différentes échelles de temps : les sous-tâches sont des problèmes de moindre envergure, mais de plus fine granularité, et une fois ces sous-tâches maîtrisées, c'est à leur "assemblage" qu'il faut s'intéresser.

Les approches apparentées aux SMDP de Sutton [Precup et Sutton, 97; Sutton *et al.*, 1998; McGovern *et al.*, 1998] sont des exemples d'approches de ce type. On peut aussi citer le formalisme MAXQ de [Dietterich, 2000] (voir figure 5.1) ou l'utilisation de sous-espaces d'états faite dans [Drummond, 2002] par exemple (des politiques étant connues pour des salles types, ce qui permet de planifier à l'intérieur d'un bâtiment complet).



Ici, un taxi doit s'occuper de clients allant d'une des positions marquées R(ouge), V(ert), J(aune) ou B(leu) à une autre. Les actions primitives correspondent aux 4 déplacements élémentaires ainsi qu'à `charger` et `déposer`. Les récompenses sont de -1 à chaque déplacement, de -10 pour toute prise en charge ou toute dépose inutile, et +20 pour un service accompli.
Les deux principales sous-tâches correspondent à `aller prendre` et `aller déposer` un client, chacune nécessitant la capacité d'aller à l'une des 4 positions puis d'exécuter une action `charger` ou `déposer`.

FIG. 5.1 – Problème du taxi tiré de [Dietterich, 2000].

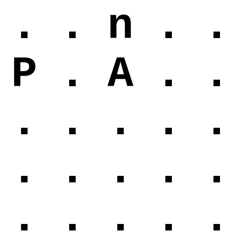
Dans les formes **"concurrentes"**, plusieurs contrôleurs (associés chacun à un PDM) sont présents simultanément. Ces contrôleurs peuvent correspondre à des objectifs distincts, comme c'est généralement le cas, ou à diverses manières d'atteindre un même but ([Scherrer, 2003] met ainsi en œuvre des PDM qui se spécialisent chacun sur une partie de l'espace d'états). Quoi qu'il en soit, il faut alors gérer leurs interactions, soit en construisant un PDM unique sur la base des connaissances des sous-PDM (voir [Dixon *et al.*, 2000] ou [Singh et Cohn, 1998], ce dernier étant illustré par la figure 5.2), soit en se limitant à une heuristique qui "élit" l'action à effectuer.

La distinction entre ces deux domaines n'est en fin de compte qu'une distinction de principe entre un cas où les agents attendent la terminaison de l'exécution de l'agent en activité et un autre cas où les agents sont constamment en compétition. Elle correspond surtout à des besoins différents, mais les principes de fonctionnement restent assez proches.

Une dernière remarque est que, si l'on n'a évoqué pour l'instant que des architectures à un seul niveau d'abstraction, il faut garder à l'esprit qu'il peut y en avoir un plus grand nombre, formant ainsi des hiérarchies assez complexes.

5.2.3 Difficultés

Un premier problème crucial dans toutes ces approches est la définition même de la décomposition qui va être employée. Malheureusement, la plupart d'entre elles se limite à une mise



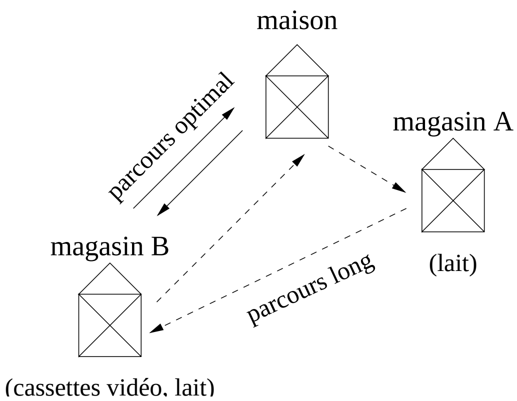
L'agent (A) erre dans une grille n par n. Il obtient une récompense de $0,5$ à chaque pas de temps où il échappe au prédateur (P), et gagne une récompense de $1,0$ à chaque morceau de nourriture (n) qu'il trouve. Quand de la nourriture est trouvée, elle réapparaît à une position au hasard au prochain pas de temps. A chaque pas de temps, S a 10% de chances d'ignorer sa politique est de faire un mouvement aléatoire.
L'algorithme présenté dans [Singh et Cohn, 1998] "fusionne" les comportements assez facilement appris pour les deux tâches élémentaires *évite-le-prédateur* et *mange-la-nourriture*.

FIG. 5.2 – Problème agent-prédateur-nourriture tiré de [Singh et Cohn, 1998].

au point manuelle par un concepteur (qui va donc définir les sous-tâches à apprendre). Seuls quelques travaux tels que ceux de [Laurent, 2002] essaient de se passer d'une telle intervention (en l'occurrence en utilisant des apprentissages parallèles).

Un autre aspect qui pose des difficultés est le fait que, selon le niveau d'abstraction auquel on se place (dans la hiérarchie de politiques de l'agent), l'espace d'observation peut lui aussi être plus ou moins abstrait, ne retenir que les éléments utiles au choix du prochain sous-comportement à adopter par exemple. Mais ici encore, la détermination de l'espace d'observation à employer reste le domaine d'un concepteur humain.

Pour terminer, insistons sur un défaut inhérent à l'approche par décomposition. A moins de se trouver dans des situations particulières assez idéales, on risque très souvent de ne plus être capable de trouver une solution optimale. En effet, si chaque sous-PDM peut certes être résolu optimalement, la recomposition en un comportement "global" n'est pas toujours bonne. L'exemple fourni par [Dietterich, 2000] et reproduit à travers la figure 5.3 illustre assez bien ce point.



Si l'on veut acheter du lait *et* une cassette vidéo, aller acheter du lait au magasin le plus proche (ici A) *et* aller acheter une cassette après (nécessairement en B) peut constituer une mauvaise solution si les deux achats peuvent être effectués dans un seul et même établissement (B dans le cas présent).

FIG. 5.3 – Problème de l'optimisation de courses tiré de [Dietterich, 2000].

Point de vue

Comme on l'a dit en introduction de cette troisième partie du manuscrit, nous avons pour notre part décidé de porter notre intérêt vers une décomposition en *tâches concurrentes*. Mais cette problématique se rapproche aussi du domaine de la sélection d'action dont il va être maintenant sujet.

5.3 Sélection d'action

Dans la présentation des approches "concurrentes" (section 5.2.2), nous avons évoqué sans plus de détails la possibilité d'utiliser une heuristique qui "élit" l'action à effectuer en observant les choix que feraient des contrôleurs chacun guidé par son propre but. Le problème soulevé ici entre dans le champ de ce que l'on appelle la "sélection d'action" (notée S/A). Nous allons présenter ici ce domaine, en profiter pour décrire le problème qui nous servira de référence dans cette troisième partie du présent mémoire (le monde des tuiles), et discuter de quelques points importants en S/A.

Note : les principales sources employées ici dans le domaine de la sélection d'action sont les thèses de Tyrrell et Humphrys [Tyrrell, 1993; Humphrys, 1997], l'introduction qui suit est d'ailleurs largement inspirée des écrits de ce dernier.

5.3.1 Qu'est-ce ?

Par "sélection d'action", nous n'entendons pas le problème de bas-niveau du choix d'une action dans la poursuite d'un unique but cohérent (tel que trouver la sortie d'un labyrinthe). Nous entendons plutôt le problème de haut-niveau du choix d'une action dans le cadre de buts conflictuels et hétérogènes (voir figure 5.4). Ces objectifs sont poursuivis en parallèle. Ils peuvent parfois se combiner pour accomplir des objectifs à plus grande échelle, mais généralement ils interfèrent simplement entre eux. Ils peuvent ne pas avoir de conditions de terminaison.

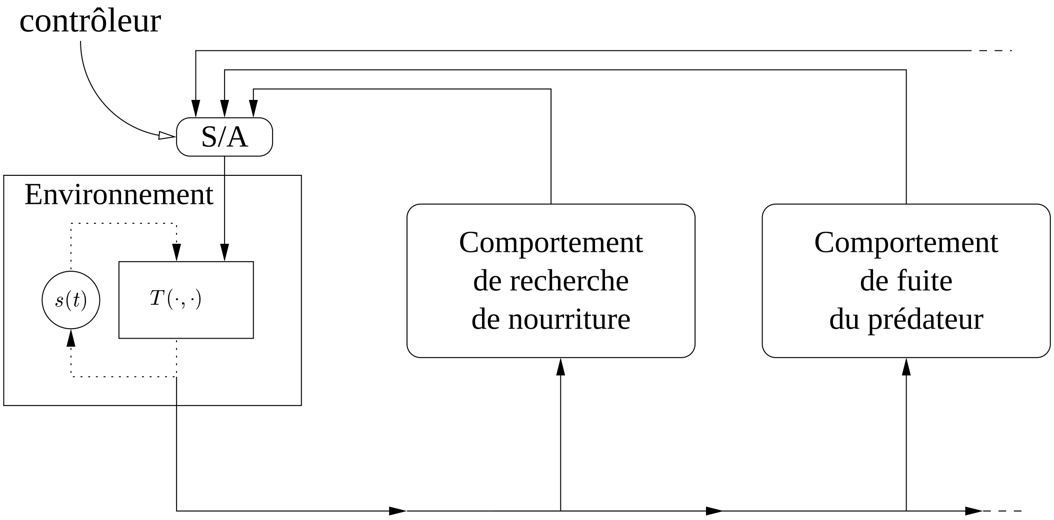
La sélection d'action apparaît comme un problème central dans la simulation de créatures complètes. Il est clair que de nombreux systèmes intéressants poursuivent de multiples buts parallèles et conflictuels, entre lesquels l'attention doit continuellement alterner (ou plutôt qu'il faut prendre en compte simultanément). Les animaux sont le premier exemple de tels systèmes.

Généralement, les modèles de sélection d'action proposés en éthologie ne sont pas assez détaillés pour spécifier une implémentation algorithmique (voir [Tyrrell, 1993] pour un aperçu, et pour les nombreuses difficultés de la traduction de modèles conceptuels en modèles computationnels). Les modèles de sélection d'action qui se prêtent à des implémentations algorithmiques (voir [Brooks, 1991b; Rosenblatt, 1995; Blumberg, 1994; Sahota, 1994; Aylett, 1995]) requièrement généralement un effort de conception considérable. Dans la littérature, on voit des formules utilisant des sommes pondérées de quantités diverses dans une tentative d'estimer l'utilité des actions. Il y a beaucoup de mises au point et réglages manuels de paramètres (voir [Tyrrell, 1993, chapitre 9]) avant que le concepteur soit satisfait et que les estimations d'utilité délivrées par les formules soient assez bonnes.

5.3.2 Le monde des tuiles

Après l'exemple de la figure 5.2, nous présentons ici en second exemple de problème de sélection d'action celui du *monde des tuiles*[39] qui va être utilisé dans les chapitres suivants pour

[39] A ne pas confondre avec le "monde des blocs".



Le problème agent-prédateur-nourriture de la figure 5.2 est typiquement un problème de sélection d'action. Si l'agent sait d'une part éviter le prédateur et d'autre part se nourrir, poursuivre ces deux buts parallèlement pose problème si le prédateur est entre l'agent et sa nourriture (c'est là que les objectifs interfèrent : ils risquent de se contredire quant au déplacement à effectuer).

FIG. 5.4 – Le problème agent-prédateur-nourriture vu comme un problème de sélection d'action.

illustrer les idées que nous proposons et les résultats obtenus. Il faut noter que la définition de ce problème n'est pas unique (voir aussi [Joslin *et al.*, 1993; Wooldridge *et al.*, 1995]). Nous détaillons ici la version mono-agent que nous avons employé.

5.3.2.1 Problème

Le monde des tuiles est un environnement quadrillé dans lequel une case peut contenir un *trou*, une *tuile* ou un *agent*. Dans le problème complet, l'agent (on en considère ici un seul) doit pousser des tuiles dans des trous aussi souvent que possible, tout en évitant de passer lui-même dans l'un de ces trous.

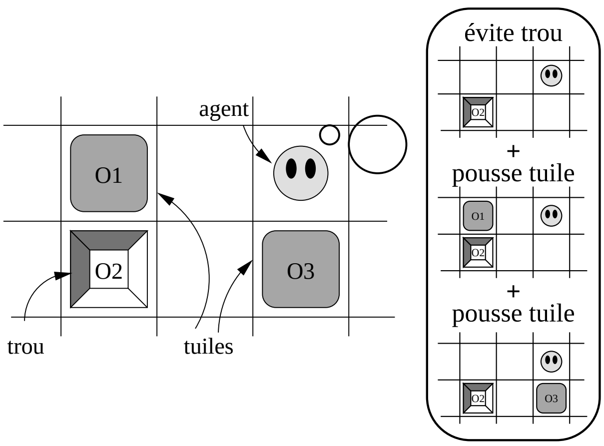


FIG. 5.5 – Une scène avec quelques objets : la tâche globale est une combinaison de sous-tâches.

Quelques détails sur la simulation ont leur importance. D'abord, l'agent peut aller librement dans un trou (et aussi en sortir), mais recevra alors une récompense négative. Ainsi le fait de tomber dans un trou n'est pas mortel. De plus, quand une tuile est poussée dans un trou, tuile *et* trou disparaissent et réapparaissent au hasard quelque part sur la grille, ce qui amène l'environnement à changer aléatoirement au cours du temps. Finalement, pour éviter quelques situations de blocage, tuiles et trous ne peuvent être sur des cases en bord du quadrillage (comme c'était le cas dans le problème de fusion de bloc en section 4.2.2.1).

5.3.2.2 Caractéristiques de l'agent

Les caractéristiques de l'agent qui vont maintenant être décrites ressemblent fortement à celles qui ont été présentées en section 4.2.2.1, ce qui va amener à redire certaines remarques.

perceptions : Dans nos travaux, l'agent a toujours tous les autres objets de l'environnement en vue. Le principe de localité est toutefois présent dans le fait que les perceptions sont imprécises. Pour tout objet O de la scène, la perception que l'agent a de O donne :
- `dir(`O`)` : Sa direction `(N-NO-O-SO-S-SE-E-NE)`.
- `proche(`O`)` : Est-il dans le carré de 9 cases centré sur l'agent `(vrai|faux)` ?

Ainsi, pour chaque objet on a $2 * 8 = 16$ perceptions possibles, d'où un espace de taille au pire 16^n si n objets sont présents. Encore une fois, des symétries et situations impossibles permettent de réduire cette dimension. Mais cela dépendra de la nature des différents objets (on ne peut considérer comme équivalents que des objets de même nature : 2 tuiles par exemple).

actions : Les seules actions disponibles pour l'agent sont de se déplacer d'une case vers le `nord`, le `sud`, l'`est` ou l'`ouest`, poussant éventuellement une ou plusieurs tuiles. Avec ce choix, l'agent ne peut tenter de rester sur sa case actuelle, ce qui de toute manière lui ferait perdre du temps.

Cette fois, comme on n'est en présence que d'un seul agent, il n'y a plus de conflits dans les actions possibles. On se retrouve avec une évolution déterministe (aux apparitions d'objets près), n'ayant pas bruité les déplacements.

récompenses : Pour conclure, la récompense reçue est de :

(+1) quand une tuile tombe dans un trou,

(−3) quand l'agent passe dans un trou, et

(0) le reste du temps.

5.3.2.3 La sélection d'action dans ce problème

Dans ce problème complexe complet, de nombreuses tuiles et trous doivent être gérés. Comme on peut le voir sur la figure 5.5, une décomposition simple du problème en comportements élémentaires (on parlera de *comportements de base*) peut être faite intuitivement, en deux comportements dans ce cas précis[40] :
- `[évite trou]`, qui met en jeu un trou, ce que l'on note $(b_e, \langle trou \rangle)$ et
- `[pousse tuile dans trou]`, qui met en jeu à la fois une tuile et un trou, ce que l'on note $(b_p, \langle tuile,\ trou \rangle)$.

Ces deux comportements (ou plus précisément les politiques associées) s'apprennent très bien séparément l'un de l'autre, il suffit pour cela de définir les fonctions de récompense associées (ce

[40] Pour des raisons pratiques, on note par la lettre b (pour "behavior") un comportement.

qui ne pose pas de problèmes) et laisser l'apprentissage se faire, avec la montée de gradient de Baxter vue en section 2.3.3.2 dans notre cas.

Il reste par contre à voir comment recombiner ces comportements, en notant au passage que, si l'environnement contient par exemple deux trous ou plus, plusieurs occurrences (on parlera d'"instances") du même comportement sont à prendre en compte : n_{trous} instances de `[évite trou]` et $n_{trous} * n_{tuiles}$ instances de `[pousse tuile dans trou]`. Dans l'exemple de la figure 5.5, on a ainsi :

- 1 instance de `[évite trou]` : $(b_e, \langle O_2 \rangle)$, et
- 2 instances de `[pousse tuile dans trou]` : $(b_p, \langle O_1,\ O_2 \rangle)$ et $(b_p, \langle O_3,\ O_2 \rangle)$.

5.3.3 Le gagnant prend tout

Dans le domaine de la sélection d'action, on peut généralement présenter les algorithmes utilisés comme des assemblées d'agents représentant chacun la politique associée à un comportement. Le problème d'une telle assemblée est d'opter pour une des actions possibles, et donc quel mode de scrutin adopter.

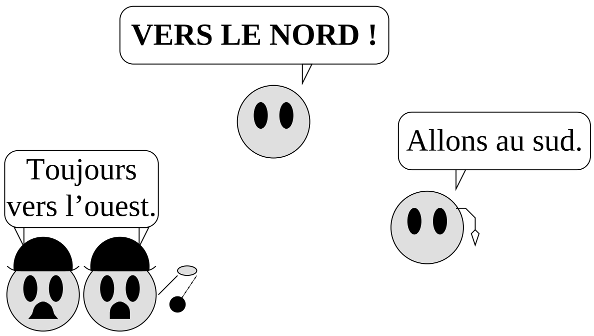


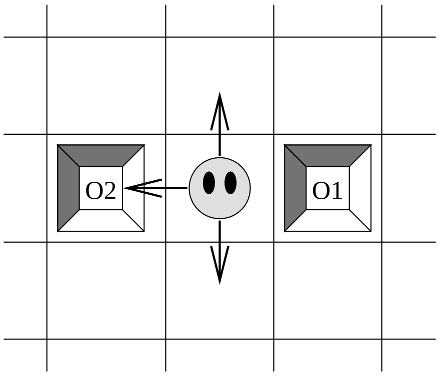
FIG. 5.6 – Une assemblée d'agents d'avis divers.

Une première manière de prendre une décision (souvent rencontrée) est de choisir, sur la base d'un critère donné, *un* des comportements, et de le mettre en application comme s'il était seul. C'est ce que l'on appelle un algorithme *winner-take-all*. Sur ce principe, Tyrrell décrit (dans [Tyrrell, 1993]) les systèmes de sélection d'action organisés en *structure de décision hiérarchique* (ou *hierarchical decision structure*). Il s'agit de systèmes, organisés donc de manière hiérarchique, dans lesquels une décision est prise à chaque niveau quant au choix du sous-système à activer.

Une telle approche appliquée au choix d'une direction à suivre schématisé sur la figure 5.6 correspondrait à n'écouter que l'agent qui parle le plus fort : "Vers le nord !" dans ce cas précis.

5.3.3.1 Raisons pour trouver une autre approche

Comme l'exemple qui vient d'être donné peut déjà le faire sentir, l'approche winner-take-all est en générale condamnée à échouer dans certaines situations qui requièrent réellement un compromis. Sur la figure 5.7, un agent doit éviter deux trous. Ainsi, deux instances du comportement b_e d'évitement de trou (l'un considérant O_1, l'autre O_2) expriment leurs préférences, chacun disant en fait de ne pas aller dans la direction du trou auquel il doit faire attention (c'est possible puisqu'on considère des comportements stochastiques). Comme il ne doit y avoir qu'un gagnant, seul un des deux comportements est suivi, évitant O_1 par exemple. Ceci conduit à un choix équiprobable de déplacement dans l'une des trois directions autorisées (nord, sud ou ouest), et donc à une chance sur trois de tomber dans l'autre trou O_2.



Avec un algorithme winner-take-all, l'agent a une chance sur deux de ne se soucier que du trou O_1, et ainsi d'obéir à une politique le faisant aller dans n'importe quelle direction parmi nord, sud et ouest. Dans ce dernier cas, l'agent va chuter dans l'autre trou.

FIG. 5.7 – Un agent devant éviter deux trous.

S'il n'est pas prouvé qu'une autre combinaison de comportements de base puisse produire un comportement complexe optimal, un procédé de type winner-take-all est vraisemblablement loin de faire un usage satisfaisant des informations fournies par les comportements de base.

5.3.3.2 Flux-libre

Plutôt que d'employer une structure de décision hiérarchique, une autre option est de combiner les politiques de base en une *nouvelle* politique. Cette variante est plus attrayante parce qu'elle permet de faire les compromis appropriés pour résoudre des problèmes tels que montrés sur la figure 5.7, où les meilleures décisions à prendre ne sont données par aucune sous-politique, mais peuvent émerger d'une politique combinée.

Calculer ainsi une nouvelle politique peut être vu comme une *hiérarchie à flux libre* (ou *free-flow hierarchy*) telle que définie par [Tyrrell, 1993]. Il s'agit de mettre en œuvre un mécanisme d'évaluation de chaque action en tenant compte des intérêts de chacun, une dernière étape consistant à passer de cette évaluation à une prise de décision.

Le lecteur intéressé par une argumentation plus complète en faveur des hiérarchies à flux libre plutôt que des structures de décision hiérarchiques pourra se reporter à la thèse de Tyrrell, dans laquelle la discussion est fondée sur l'étude approfondie d'un certain nombre de travaux dans le domaine de la sélection d'action.

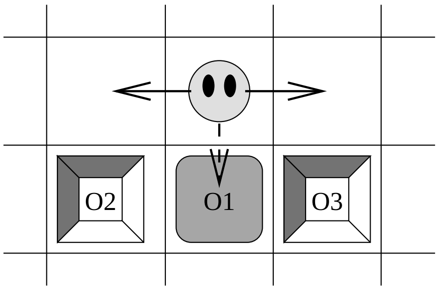
Note : Comme on le voit dans cette discussion sur les architectures possibles de sélection d'action, on se retrouve devant le choix d'une architecture interne d'agent (voir section 1.1.5.3), point essentiel de la conception d'un agent, comme cela avait été souligné en section 1.3.1.1.

5.3.4 Politique stochastique ou déterministe ?

Un point que nous évoquerons ici succinctement est le fait que la plupart des algorithmes de sélection d'action rencontrés se contentent de produire des politiques déterministes, ne proposant qu'une action par situation rencontrée, alors que les cadres de travail sont souvent non-markoviens, et en tout cas ne garantissent pas qu'une politique déterministe soit efficace.

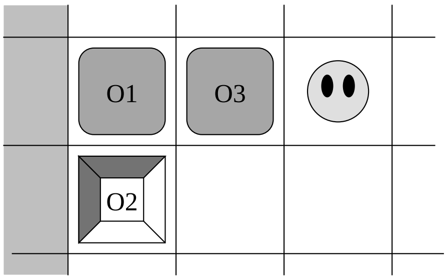
Les figures 5.8 et 5.9 illustrent un autre argument pour l'utilisation de politiques stochastiques qui soient “bruitées” : dans l'un comme l'autre cas, aucune des actions qui peuvent être proposées par l'un des comportements de base impliqués n'amène à résoudre le problème. Mais nous retrouverons cette difficulté ultèrieurement.

Comme nous l'avons fait jusqu'ici, nous chercherons donc à employer des méthodes de manipulation de politiques stochastiques.



Ici, deux contrôleurs sont prépondérants (momentanément) pour l'agent :
– `pousser` O_1 dans O_2 qui incite à aller vers l'est, et
– `pousser` O_1 dans O_3 qui incite à aller vers l'ouest.
Or le choix le plus judicieux serait plutôt le sud, pour sortir la tuile d'entre les deux trous.

FIG. 5.8 – Blocage avec 1 tuile et 2 trous.



Que ce soit pour `pousser` O_1 dans O_2 ou `pousser` O_3 dans O_2, les deux contrôleurs prépondérant incitent à aller vers l'ouest, ce qui ne permet pas au système d'évoluer.
Mieux vaudrait aller vers le nord pour aborder l'une des tuiles par un autre côté.

FIG. 5.9 – Blocage avec 2 tuiles et 1 trou (et des cases interdites à l'ouest).

5.3.5 Bilan intermédiaire

Comme la définition qui en a été donné en section 5.3 le laisse déjà apparaître le problème de la sélection d'action est très ouvert. Il ne représente pas un ensemble de techniques et méthodes précises comme c'est le cas par exemple des PDM, mais correspond à une assez grande variété d'approches. Nous en avons donné ici un premier aperçu qui suffira à notre propos dans les sections et chapitres qui suivent. Pour ne donner que cet exemple, nous avons omis les mécanismes non-hiérarchiques tels que ceux proposés dans [Maes, 1989] (il s'agit d'un réseau de modules qui peuvent s'inhiber ou s'activer mutuellement, ce qui ressemble à l'architecture à subsumption de Brooks décrite en section 1.1.5.3).

Pour l'instant, nous avons exprimé le choix de travailler sur des architectures à flux-libre, et en produisant des politiques stochastiques. Nous allons voir maintenant le lien qui peut être fait avec le domaine de l'apprentissage par renforcement qui reste la base de nos travaux.

5.4 Sélection d'action et apprentissage par renforcement

Les sections 5.2 et 5.3 ont permis de présenter d'une part les travaux visant à décomposer des processus de décision markoviens pour en rendre le traîtement plus efficace, et d'autre part le domaine de la sélection d'action. Le sujet qui nous motive dans ce chapitre est à l'intersection de ces deux domaines. Nous allons donc voir à présent ce qui a déjà été fait sur ce thème alliant A/R et S/A, en ayant comme principale référence la thèse de Humphrys, laquelle est centrée sur ce sujet.

5.4.1 Exemple : W-learning

Une fois n'est pas coutume, commençons par un exemple. Les travaux présentés dans [Humphrys, 1996] et développés de manière plus complète dans [Humphrys, 1997] présentent (séparément) les deux approches dont il a été sujet en section 5.3.3 : *avec* et *sans* winner-take-all.

Tous les algorithmes présentés emploient un contrôleur nourri non seulement des actions souhaitées par chaque comportement, mais aussi d'informations sur l'utilité de ces actions. Différentes stratégies ont été étudiées pour effectuer la sélection d'action. Elles sont présentées en deux catégories (on note ici Q_i les Q-valeurs de l'agent-comportement i, et a_i la meilleure action pour le même agent) :

1. les W-learnings individualistes (*avec* winner-take-all : *un* des agents-comportements est maître de la décision) :
   - maximiser la plus grande joie
   $$\max_{a \in \mathcal{A}} \max_{i \in 1,\ldots,n} Q_i(s,a), \text{ et} \tag{5.1}$$
   - minimiser le plus grand regret
   $$\min_{a \in \mathcal{A}} \max_{i \in 1,\ldots,n} [Q_i(s,a_i) - Q_i(s,a)]; \text{ et} \tag{5.2}$$
2. les W-learnings collectifs (*sans* winner-take-all : la décision vient d'un concensus entre les agents-comportements) :
   - maximiser la joie collective
   $$\max_{a \in \mathcal{A}} \sum_{i=1}^{n} Q_i(s,a), \text{ et} \tag{5.3}$$
   - minimiser le regret collectif
   $$\min_{a \in \mathcal{A}} \sum_{i=1}^{n} [Q_i(s,a_i) - Q_i(s,a)]. \tag{5.4}$$

Note : Si les secondes approches sont de type flux-libre, elles ont le défaut signalé en section 5.3.4 de fournir des politiques déterministes.

L'intérêt des travaux de Humphrys réside dans le fait que les poids à donner aux différents comportements sont appris, ce qui permet d'ajuster la combinaison effectuée. Toutefois, même si le but était ici que les agents puissent se passer de toute fonction de récompense globale, il a fallut qu'une adaptation des fonctions de récompense locales (à chaque comportement de base) soit faite (à l'aide d'un algorithme génétique dans lequel la récompense globale ré-apparaît). Ainsi, cette adaptation s'avère très spécifique à la situation courante.

Le procédé auto-adaptatif utilisé dans le mécanisme de sélection d'action encourage toutefois à employer des méthodes de combinaison dans lesquelles les paramètres puissent être appris par renforcement.

5.4.2 Problème posé

On rejoint en fait ici la problématique de la décomposition d'un PDM en sous-tâches concurrentes (introduite en section 5.2.2), ce qui transparaît dans le travail de Humphrys. A partir de sa lecture, nous allons essayer ici de préciser l'objectif que nous visons, ainsi que la manière de l'atteindre.

En effet, Humphrys cherche justement dans sa thèse [Humphrys, 1997] à mettre au point une méthode pour organiser automatiquement la sélection d'actions sur la base de comportements élémentaires. Il choisit pour cela (comme on vient de le voir dans la section précédente) d'utiliser des méthodes d'apprentissage par renforcement, mais fait aussi référence à des travaux issu plus directement du champ des processus de décision markoviens. Il cite ainsi dans le chapitre 3 des méthodes d'apprentissage par renforcement multi-modules [Moore, 1990; Singh, 1992; Tham et Prager, 1994] (ces deux derniers concernant des approches séquentielles et non concurrentes), et plus particulièrement le Q-learning hiérarchique de [Lin, 1993]. Ce dernier travail est assez typique du domaine, puisqu'il s'agit d'apprendre par Q-learning :

- d'abord les n sous-tâches définies (chacune associée à un "agent" A_i),
- puis un contrôleur dont l'action soit de choisir l'agent à utiliser (dans chaque état possible du système).

La lecture de la thèse de Humphrys montre une certaine diversité des approches rencontrées, entre autres du fait d'objectifs et de cadres divers. Il est ainsi difficile d'en tirer de réelles conclusions. C'est d'autant plus vrai de notre point de vue, car tous se basent sur le formalisme des PDM, alors que l'hypothèse de Markov devient irréaliste dans les cas auxquels nous souhaitons nous attacher.

Mais, de manière générale, on peut extraire un principe assez simple, celui d'apprendre séparément à résoudre des tâches indépendantes, puis d'apprendre un contrôleur pour les recombiner. La forme du contrôleur à employer est souvent le principal sujet de discussion. C'est d'ailleurs le cas dès qu'il s'agit de sélection d'action, mais ici s'ajoute le fait que des paramètres viennent prendre place et qu'un algorithme pour les optimiser (les apprendre) doit être mis au point.

En plus de ces aspects, auxquels nous n'échapperons pas en proposant notre approche, nous essaierons de déterminer de manière automatique quelles sont les sous-tâches à savoir accomplir, ou plus précisément les comportements de base à utiliser.

5.5 Est-ce bien SMA ?

Avant de clore ce chapitre, revenons à son intitulé : "SMA pour PDM".

Aucun système multi-agents au sens qu'on a donné à ce terme dans la section 1.2 n'a été ici évoqué, puisqu'on a toujours supposé distinctes une phase de conception de "modules" élémentaires et une phase d'utilisation combinée de ces modules (dès lors fixés), alors qu'un agent devrait toujours être capable de s'adapter. Notons qu'il existe quelques travaux, tels que ceux de [Laurent, 2002], qui cherchent à faire un apprentissage en parallèle et *en situation* de comportements élémentaires. Il est donc envisageable de pallier ce défaut.

Un autre aspect qui peut rendre le terme SMA discutable est qu'il y a toujours un mécanisme centralisateur reliant les différents agents élémentaires au sein de l'agent "englobant" cette asssemblée. Si l'on se place du point de vue de ses agents parlementaires, le mécanisme centralisateur fait toutefois partie de l'environnement. En revenant au modèle influences-réaction vu en section 1.2.1 (page 28), les actions des agents vont correspondre aux influences, et le mécanisme centralisateur sera en grande partie l'opérateur Π de combinaison de ces influences. La figure 5.10 illustre ce lien en prenant l'exemple d'un agent dans le monde des tuiles.

Pour dire un dernier mot de la question "Peut-on parler de SMA ?", on retrouve dans ce principe de décomposition d'un agent en sous-agents l'idée de Minsky [Minsky, 1986] d'une "société de l'esprit", même si sa vision concerne un système plus général et plus complexe. Cette référence se rencontre d'ailleurs assez régulièrement dans les écrits sur la sélection d'action.

Ayant conscience que le qualificatif de SMA reste discutable, et surtout dans le but de ne pas être ambigu en parlant d'agent, nous parlerons de préférence de comportements de base (ou élémentaires) plutôt de d'agents ou sous-agents, le mot "agent" étant réservé au seul système complexe qu'on cherche à réaliser.

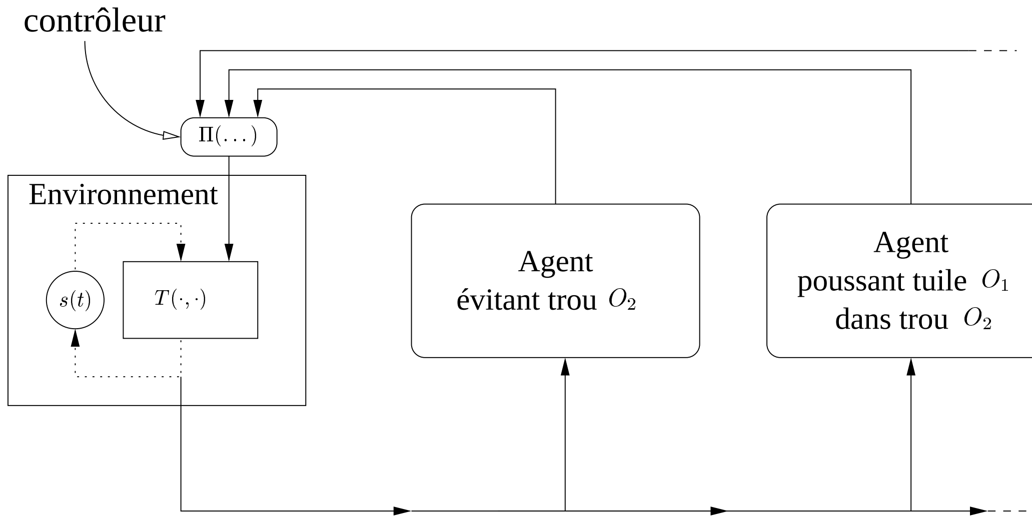


FIG. 5.10 – Forme d'un mécanisme de sélection d'action d'après le point de vue du modèle influences-réaction (fait référence à la figure 1.10).

Ce qui manque principalement à l'architecture de sélection d'action que nous allons proposer pour pouvoir être réellement considérée comme un SMA est une véritable forme *adaptative*. Mais nous gardons pour des travaux futurs l'idée d'élaborer un véritable contrôleur à géométrie variable, moins centralisateur que celui choisi dans la présente thèse.

5.6 Conclusion

Nous avons rappelé dans ce chapitre qu'un processus de décision markovien (et même non-markovien à vrai dire) se trouve facilement confronté à des difficultés liées à l'explosion de l'espace d'observation. Partant du principe de "diviser pour régner", différentes approches de décomposition d'un PDM ont été exposées (les cas non-markoviens étant rarement discutés).

Parmi ces approches, nous proposons de nous intéresser à celles qui mettent en œuvre des tâches concurrentes, ce qui amène naturellement à examiner les travaux du domaine de la sélection d'action. Sur la base de tout cela, il a été décidé de chercher à concevoir une approche de sélection d'action de type *hiérarchie à flux libre*, produisant une *politique stochastique*, la mise au point étant faite par *apprentissage par renforcement* autant pour les comportements de base que pour le contrôleur. C'est le sujet du chapitre 6 qui suit.

De plus, on a souhaité *automatiser le choix des sous-tâches* à savoir résoudre. Dans ce sens, le chapitre 7 commence par montrer comment créer un nouveau comportement à partir des ceux déjà connus, ce procédé étant employé dans l'algorithme proposé au chapitre 8 pour effectivement déterminer les comportements de base à utiliser.

Chapitre 6

Simple combinaison de comportements

Sommaire

La discussion sur la décomposition de PDM et sur le domaine de la sélection d'action qui a été menée dans le chapitre précédent (chapitre 5) a conduit à vouloir réaliser un mécanisme de sélection d'action de type hiérarchie à flux libre, et qui permette d'obtenir une politique stochastique. On reste ici dans l'optique d'un agent sans mémoire et avec perceptions partielles, en cherchant à automatiser sa conception à travers un apprentissage par renforcement.

Le présent chapitre va proposer une méthode de sélection d'action adaptative que nous avons élaborée de manière à ce qu'elle réponde à nos vœux. Les deux premières sections présentent les deux principales phases du mécanisme qui va être réalisé :

1. En guise d'introduction, la section 6.1 présentera le principe de décomposition d'une tâche (et donc d'analyse de la scène courante, ce qui constitue la première phase de l'algorithme) qui sera adopté.
2. Ceci permettra de poursuivre dans la section 6.2 en proposant pour la seconde phase, de "recomposition", une méthode générique de combinaison de comportements, ainsi que les formes spécifiques qui ont été étudiées.

Les expériences menées et leurs résultats sont alors présentés en section 6.3, avant évidemment de faire un bilan (section 6.4).

6.1 Principe de la décomposition

Dans un premier temps, nous allons voir à la fois comment une tâche complexe est décomposée en sous-tâches, et comment est faite l'analyse des perceptions de l'agent pour déterminer les sous-buts courants. Ces deux aspects sont en réalité très liés l'un à l'autre, ce qui fait qu'on ne les distinguera pas clairement.

On verra aussi dans cette section l'algorithme que nous proposons dans ses grandes lignes (à la fois les phases de décomposition et de combinaison), de manière à introduire justement les détails de l'approche, et la combinaison elle-même, dans la section suivante.

6.1.1 Comment une tâche est décomposée

Nous introduisons ici un formalisme assez précis pour décrire ce que perçoit un agent, et comment l'agent analyse ses perceptions.

Ce que perçoit l'agent

Nous supposerons que l'agent considéré est dans un monde d'*objets de types définis*. L'étude complète menée ici pourrait en fait prendre directement en compte des *perceptions typées* au lieu d'objets, mais cette approche objet sera préférée pour simplifier la compréhension.

Ainsi, l'agent perçoit la scène courante (la partie accessible de son environnement) comme une **observation**, ensemble composé de **percepts**, chacun lié à un objet voisin. Il n'a donc accès qu'à une information partielle sur l'état du système. Supposant, par exemple, que la figure 6.1 ne montre que les trois objets (2 tuiles et 1 trou) visibles, l'observation de l'agent est ici constituée des percepts de chacun de ces trois objets.

Comment l'agent analyse ses perceptions

L'hypothèse principale faite dans notre approche est que la tâche d'un agent peut être vue comme la composition de tâches plus élémentaires (sous-buts), chacune n'impliquant qu'un en-

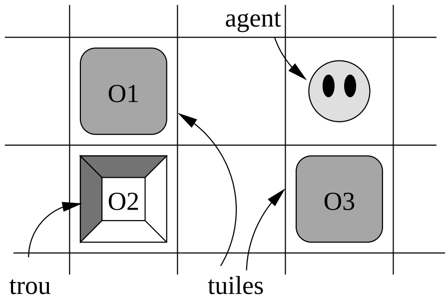


FIG. 6.1 – Une scène avec quelques objets voisins, d'autres objets peuvent être présents dans l'environnement, mais hors du champ de vision.

semble limité d'objets (ensemble appelé **configuration**), et des **comportements de base**[41] appropriés étant connus pour ces sous-tâches. En fait, comme des objets typés sont utilisés, un comportement sera défini seulement pour un tuple de types d'objets (tuple appelé **type de configuration**), comme discuté ultèrieurement en section 6.2.1. Dans la situation de la figure 6.1, un exemple de comportement de base utilisé est de *pousser* une tuile dans un trou (type de configuration $\langle tuile,\ trou\rangle$). Le comportement élémentaire *pousser* (b_p) apparaît dans ce cas avec les configurations $\langle O_1, O_2\rangle$ et $\langle O_3, O_2\rangle$.

Parmi toutes les configurations possibles dans une scène (c'est-à-dire, tous les sous-ensembles possibles d'objets), seule une part restreinte sera intéressante : celles liées à des sous-tâches. Dans notre exemple, seules $\langle O_2\rangle$, $\langle O_1, O_2\rangle$ et $\langle O_3, O_2\rangle$ sont utiles (pour *éviter* un trou (b_e) ou *pousser* une tuile dans un trou (b_p)), et pas $\langle O_1, O_3\rangle$ par exemple.

L'objectif de la décomposition de la scène sera l'identification de paires (*comportement de base*, $configuration$) (désormais notées (bb, cfg), bb venant de l'anglais "*basic behavior*") dans l'environnement accessible. L'application qui à l'observation courante associe l'ensemble des paires (bb, cfg) de la scène sera notée $\mathcal{BC}$. Ainsi, pour conclure l'analyse de la figure 6.1, avec les deux comportements de base intuitivement utilisés, l'agent déduira de la décomposition de la scène courante qu'il doit tenir compte des trois paires (bb, cfg) qui suivent :

$$\mathcal{BC}(o) \quad = \quad \{(b_e, \langle O_2\rangle),\ (b_p, \langle O_1, O_2\rangle),\ (b_p, \langle O_3, O_2\rangle)\}. \tag{6.1}$$

Scalabilité

Comme exprimé en introduction de cette troisième partie, on cherche à ce que l'agent soit capable de "se mettre à l'échelle", c'est-à-dire ici qu'il puisse gérer un nombre variable d'objets (comme expliqué en section 4.3, on parlera de "scalabilité" et d'agent "scalable"). Si l'on ne tient compte que de ce processus de décomposition d'une scène, la scalabilité vient de ce qu'un comportement de base (respectivement une configuration) peut être associé à plus d'une configuration (resp. comportement de base), puisqu'un comportement de base dépend d'un type de configuration générique. C'est d'autant plus intéressant que, du fait de la localité des perceptions de l'agent, l'ensemble des configurations utiles change selon les objets perceptibles.

De nouveaux problèmes de complexité ?

Considérant la décomposition d'une scène en paires (bb, cfg), on peut aussi noter que, parmi les points délicats de la méthode que nous proposons, une explosion combinatoire est à craindre

[41] Des définitions détaillées sur les comportements viendront en section 6.2.1.

lors de la recherche de configurations utiles. Effectivement, le nombre de configurations comprenant un trou et une tuile est $n_{trous} * n_{tuiles}$, pour ne donner qu'un exemple d'ensemble à croissance rapide. Toutefois, en pratique le nombre d'objets accessibles restera généralement restreint, puisque seules des perceptions locales sont considérées.

6.1.2 Combinaison de comportements

La présentation de la décomposition de scène a apporté le principe d'utiliser des comportements *de base* qui soient génériques (définis pour des types de configuration) et correspondent à des politiques stochastiques. Avant de pouvoir dire plus précisément ce que devrait être un comportement de base, il faut commencer par chercher le moyen de combiner ces comportements de base en une politique finale que l'agent adoptera.

Mais on peut dores et déjà esquisser le mécanisme qui va être mis en œuvre à travers l'algorithme général 5. Comme dans de nombreux algorithmes d'apprentissage par renforcement, il s'agit d'une boucle dans laquelle est fait un passage par pas de temps (pour le système discret considéré). On peut la décrire rapidement à travers quatre grandes étapes :

– Lignes 1 et 2 : obtention des observations et récompenses immédiates.
– Lignes 4 à 8 : obtention des informations utiles sur les comportements de base en jeu.
– Lignes 9 et 10 : calcul d'une politique pour la situation actuelle et exécution.
– Ligne 11 : renforcement.

Algorithme 5 Algorithme général

Entrées: Comportements de base déjà connus

1: **boucle**
2: $o \leftarrow$ nouvelle observation de la scène.
3: $r \leftarrow$ récompense immédiate de l'agent.
4: Décomposition : chercher les paires $(bb,\ cfg)$ connues $(b, c) \in \mathcal{BC}(o)$.
5: **pour tout** paire (b, c) **faire**
6: $\pi_b(o(c)) \leftarrow$ distribution de probabilité induite par (b, c).
7: $\beta_b(o(c)) \leftarrow$ autres informations si nécessaire.
8: **fin pour**
9: $\mathcal{P}(o, \cdot) \leftarrow$ distribution de probabilités obtenue en combinant les distributions $\pi_b(o(c))$ (en utilisant éventuellement $\beta_b(o(c))$).
10: $a \leftarrow$ action choisie selon $\mathcal{P}(o, \cdot)$
11: Adapter et modifier les paramètres de la combinaison (à travers un apprentissage par renforcement utilisant r) de manière à améliorer les performances.
12: **fin boucle**

6.1.3 A propos de la sélection d'action

Le problème de combinaison de comportements qui nous intéresse entre, comme on l'a vu au chapitre 5 précédent, dans le champ de la sélection d'action. Pour rappel, la direction que nous décidons de suivre est celle d'une hiérarchie à flux libre produisant une politique stochastique, hiérarchie dans laquelle on souhaite par ailleurs laisser le système apprendre à "régler" la combinaison aussi bien que possible.

Scalabilité

Comme on l'a décrite en section 6.1, la décomposition de la scène en paires (*comportement de base*, $configuration$) permet qu'il y ait un nombre variable de telles paires dans le voisinage de l'agent. Néanmoins, les méthodes de recombinaison existantes (algorithmes de sélection d'action) ne sont généralement pas scalables, cette capacité étant en fait un point très rarement abordé. Quand un comportement de base peut être appliqué plus d'une fois dans une situation donnée (le même exemple de deux trous à éviter est approprié), ce doit être précisé *a priori* dans les algorithmes que nous avons pu citer (le W-learning par exemple).

Pour sa part, notre algorithme gère cette capacité de mise à l'échelle. Dans ce but, un poids est appris pour chaque comportement de base "générique" (tel que l'*"évitement de trou"*). Ainsi, même si de nombreuses instances d'un comportement de base peuvent apparaître dans une situation donnée (plusieurs trous à éviter, liés à plusieurs paires (bb, cfg) où $bb = b_e$), elles seront combinées en n'utilisant qu'un poids *commun*.

Ayant expliqué comment décomposer une scène et étant brièvement revenus sur le problème de la sélection d'action, les points clef de l'approche que nous proposons ont été mis en avant. Ayant de surcroît fourni la forme générale de l'algorithme qui va être utilisé, nous sommes prêts à entrer plus en détails dans le formalisme et les mécanismes mis en œuvre.

6.2 Détails de la combinaison

La section 6.1 a introduit une terminologie, encore assez imprécise, concernant les comportements. Ce travail préliminaire accompli va maintenant permettre de donner des définitions formelles précisant cette terminologie. Ceci rendra alors possible la présentation de la combinaison de comportements de base ainsi que, finalement, l'étude de notre algorithme complet sous les différentes formes qu'il va prendre.

6.2.1 Définitions formelles autour des comportements

Dans la présentation du processus proposé pour décomposer une scène en paires (*comportement de base*, $configuration$), la notion de comportement apparaît plusieurs fois, particulièrement à travers le terme de "comportement de base". Nous avons maintenant tous les éléments pertinents nécessaires pour donner quelques définitions dans ce domaine, définitions qui serviront ensuite dans la description d'aspects plus techniques de nos travaux.

Définition 8 (Comportement)
Un **comportement** b[42] *est défini par un tuple* $\langle \mathcal{C}_b^T, P_b, Q_b \rangle$, *où :*

1. $\mathcal{C}_b^T$ *est un type de configuration. Il permet l'appariement du comportement avec une configuration (un ensemble d'objets) si ce dernier répond aux exigences du type de configuration en question.*
2. $P_b(o, c, a)$ *est une politique de décision stochastique apprise par renforcement. Etant donnée une configuration c appropriée, cette politique associe à l'observation de c une distribution de probabilité sur les actions :* $P_b : \mathcal{O} \times \mathcal{C}_b \times \mathcal{A} \rightarrow [0, 1]$, *où* $\mathcal{C}_b$ *est l'ensemble des configurations de type* $\mathcal{C}_b^T$.

[42] On choisit la lettre 'b' de "behavior" pour réserver le 'c' aux configurations.

3. *$Q_b(o, c, a)$ est la table de Q-valeurs de cette politique. Elle donne l'espérance de récompense décomptée*[43] *quand l'action a est choisie pour l'observation o de la configuration c. (Nous revenons sur l'usage de cette utilité dans la prochaine définition.)*

Définition 9 (Comportement de base)
Un **comportement de base** *bb est un comportement qui a été choisi, avec d'autres, pour être recombiné en comportements complexes.*

Dans le processus de recombinaison, une paire (bb, cfg) servira à donner une proposition de décision. Ainsi, pour un comportement de base donné b et une configuration associée c, la décision proposée dépend des percepts des objets de c. La figure 6.2 illustre le processus d'obtention des percepts de chaque objet d'une configuration c, ce qui fournit alors son observation, et calculant finalement la politique immédiate à suivre d'après la paire (b, c). Sur l'exemple de la poussée de la tuile O_3 dans le trou O_2, les deux percepts : O_3 `près_au_sud`[44] et O_2 `loin_à_l'ouest` seront les données amenant à suggérer les actions à accomplir.

$$\text{cfg } c = \langle O_3, O_2 \rangle \quad \begin{array}{l} \mathcal{O}(O_3) = \texttt{près\_au\_sud} \\ \\ \mathcal{O}(O_2) = \texttt{loin\_à\_l'ouest} \end{array} \quad \Bigg| \to \mathcal{O}(c) \Rightarrow [a \to P_b(\cdot|o(c))]$$

FIG. 6.2 – Comment une distribution de probabilité sur des actions est obtenue pour un comportement donné b et une configuration c ($\mathcal{O}$ est la fonction d'observation est π_b la politique associée au comportement b, c'est-à-dire la distribution sur les actions définie par $P_b(o, c, \cdot)$).

Pour ce qui est de la fonction d'utilité d'un comportement (les Q-valeurs), celle-ci sera employée dans la recombinaison de comportements de base. En effet, la connaissance des politiques de ces comportements n'est pas suffisante pour prendre des décisions efficaces quand l'agent a affaire à des motivations concurrentes. Pour les peser d'une manière ou d'une autre, donnant une plus grande priorité à un danger à éviter ou à une récompense importante en vue, nous suggérons d'évaluer la situation en utilisant les Q-valeurs, puisque celles-ci nous donneront l'espérance de gain décompté (c'est-à-dire au sens propre l'"utilité") de chaque paire configuration-action. Ces Q-valeurs peuvent être apprises simultanément avec la politique du comportement de base, comme on le redira en revenant sur ce sujet dans la section 6.2.3.1.

Définition 10 (Comportement scalable)
Un **comportement scalable** *sb est défini par un algorithme de combinaison et un ensemble de comportements de base avec les paramètres de pondération associés, ce qui donne l'ensemble de paires suivant : $\{(bb, \theta_{bb})\}$. C'est le comportement complexe d'agents que nous cherchons à obtenir, dans lequel les paramètres θ ont été appris par renforcement.*

Un point remarquable à garder à l'esprit est qu'un comportement, comme nous l'avons défini, doit être utilisé avec un ensemble donné d'objets dans le voisinage, alors qu'un comportement scalable sera adapté à des nombres variables d'objets (de certains types). La définition donnée

[43] Nous reviendrons sur ce choix en section 6.2.3.1.
[44] Voir la description du problème en section 5.3.2.

d'un comportement simple correspond à l'observation d'un nombre exact n_{obj} d'objets tels que décrits par le type de configuration du comportement. Cela peut néanmoins autoriser l'observation de moins de n_{obj} objets si le percept d'un objet peut indiquer `objet_non_perçu`, alors qu'un comportement scalable peut (idéalement) gérer une scène illimitée (pas optimalement toutefois).

Pour résumer

Être capable de gérer un nombre variable d'objets dans une tâche complexe sera l'objectif de la conception d'un comportement scalable. Celui-ci se base sur un ensemble de comportements "simples" choisis : les comportements de base.

Le problème du choix de ces comportements de base parmi les comportements connus disponibles n'est pas le sujet de ce chapitre, mais reviendra dans les chapitres suivants. La section qui vient restreint son propos à la fonction de combinaison et à l'apprentissage des paramètres.

6.2.2 Formulation générale

La combinaison de comportements de base peut être faite de diverses manières, lesquelles restent de simples heuristiques. Seules des situations particulières rendent possible une convergence prouvée vers une politique optimale, comme dans le cas étudié par [Singh et Cohn, 1998], où de multiples PDM sont joints en un PDM composite unique. Cette section va maintenant présenter la formulation générale qui a servi de point de départ à nos travaux, avant d'amener quelques remarques sur les fonctions de pondération employées, et finalement de présenter les formes spécifiques qui ont été utilisées dans nos diverses expérimentations.

Le choix d'une action étant donnée un ensemble de paires (bb, cfg) peut être fait de nombreuses manières (vote, enchères, choix aléatoire). La figure 6.3 illustre de manière très schématique cette deuxième phase, en cachant dans la boîte "Combinaison" le point critique qui nous intéresse maintenant.

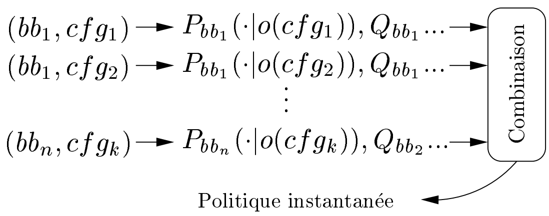


FIG. 6.3 – Schéma de la phase de recombinaison : récupération des données liées à chaque paire (bb, cfg) et production d'une politique immédiate.

Pour mettre en œuvre une hiérarchie à flux libre, nous avons décidé de calculer une politique $\mathcal{P}(o,a)$ donnant une distribution de probabilité sur les actions a pour chaque observation o. Connaissant l'ensemble des comportements de base à utiliser, la formulation générale que nous avons essayé de suivre est :

$$\mathcal{P}(o,a) \quad = \quad \frac{1}{K} \bigoplus_{(b,c)\in\mathcal{BC}(o)} w(o,b,c,a) \otimes P_b(o,c,a) \tag{6.2}$$

où les $w(o,b,c,a)$ sont des fonctions positives (appelées *poids* et dont nous reparlerons en section 6.2.3) des Q-valeurs et des paramètres θ, et où K est un facteur de normalisation (une correction pour avoir $\sum_a \mathcal{P}(o,a) = 1$). De plus, les opérateurs $(\oplus, \otimes)$ peuvent être aussi bien $(+, *)$ ou $(*, x^y)$, ou encore $(\max, +)$[45]. Une fois ces probabilités calculées pour une observation donnée, l'action est choisie selon la distribution $\mathcal{P}$.

Par la suite, on fera aussi référence à la fonction de combinaison en écrivant $\mathcal{P} = f_{combi}(\mathcal{B})$ où $\mathcal{B}$ est l'ensemble des comportements de base. Les différentes formes de f_{combi} seront présentées en section 6.2.4 dans un ordre correspondant à une évolution globalement progressive vers des traitements des données disponibles qui sont de plus en plus efficaces et adaptés.

Avant d'approfondir la question des fonctions de pondération, et comme elles seront directement concernées par l'apprentissage par renforcement (la section 6.2.3.3 reviendra sur son utilisation), nous devons préciser dès à présent la mesure de performance employée, laquelle est commune à toutes les formes. L'efficacité d'un comportement complexe est ici donnée par le renforcement moyen gagné pendant un pas de temps. Ce critère intuitif est aussi simple à évaluer. En effet, dans nos expérimentations on pourra directement, et de manière équivalente, donner le chiffre des récompenses accumulées pendant 100 000 pas de temps.

6.2.3 Réflexions sur les poids

Les différentes formes de f_{combi} employées ne peuvent être décrites sans s'être au préalable intéressé aux fonctions utilisées comme *poids* équilibrant l'importance relative des distributions de probabilité entrant en compte dans la combinaison. Deux rôles principaux vont apparaître, correspondant :

- aux Q-valeurs (section 6.2.3.1), une table étant liée à chaque comportement, et fixée *avant* d'entrer dans le processus de combinaison ; et
- aux paramètres θ, dédiés chacun à un comportement de base (section 6.2.3.2), et appris *pendant* le processus d'adaptation de la combinaison.

Toutefois, toutes les formes de f_{combi} n'utiliseront pas ces deux termes.

6.2.3.1 Utilisation des Q-valeurs dans les poids

Nous avons déjà mentionné l'intérêt des Q-valeurs dans la définition des comportements de base (voir section 6.2.1). Dans le cadre des PDM, elles donnent une grande quantité d'informations. Néanmoins, différentes formes de Q-valeurs existent, chacune ayant sa propre interprétation. Ainsi, la question qui se pose maintenant est de savoir quelle forme d'utilité choisir ? Nous allons revenir ici sur trois définitions possibles déjà rencontrées au chapitre 2, et en profiterons pour donner des arguments pour ou contre leur emploi dans la perspective d'une recombinaison.

Note : cette fois-ci, on a préféré homogénéiser les notations entre les trois définitions, et ce afin qu'une comparaison directe soit possible. De plus, l'ordre des trois présentations est différent par rapport à celui adopté au chapitre 2, progressant vers la définition ici la plus intéressante.

Récompense moyenne

$$Q^\pi(o,a) = \lim_{h\to\infty} \frac{1}{h} E^\pi \left[\sum_{k=1}^{h} r_k | o_0 = o, a_0 = a \right] \qquad (6.3)$$

L'espérance de gain moyen d'une politique π est un critère d'optimalité très intuitif. Pourtant, son usage pour définir des Q-valeurs n'a un sens réel que pour des PDM à horizon fini. Dans les

[45] Seule cette dernière paire d'opérateurs n'a pas été employée.

autres cas, tels que le nôtre, il ne donne aucune information propre aux paires observation-action, puisque toutes les observations ont la même valeur, en l'occurence l'espérance de gain moyen à long terme (dans le cas d'un unique régime stationnaire possible) :

$$\forall (o,a) \in \mathcal{O} \times \mathcal{A},\ Q^\pi(o,a) = V^\pi. \tag{6.4}$$

Cette forme de Q-valeurs ne permet dans notre situation aucune discrimination entre décisions.

Distance à la récompense moyenne

$$Q^\pi(o,a) \quad = \quad \lim_{h\to\infty} E^\pi \left[\sum_{k=1}^{h} r_k - R | o_0 = o, a_0 = a \right] \tag{6.5}$$

où

$$R \quad = \quad \lim_{h\to\infty} \frac{1}{h} E \left[\sum_{k=1}^{h} r_k \right]. \tag{6.6}$$

Dans [Jaakkola *et al.*, 1994a], comme dans [Baxter et Bartlett, 2001], l'algorithme proposé apprend une politique stochastique optimisant une récompense moyenne à long terme ($E[r]$). Mais cet article présente une autre définition des Q-valeurs (originalement introduite dans [Bertsekas, 1987]). $Q^\pi(o,a)$ peut ici être décrit comme l'espérance de "gain additionnel" à court terme (éventuellement négatif) "par rapport au gain moyen à long terme" quand l'action a est choisie pour l'observation o.

Comme cette utilité est l'outil principal dans un cadre de processus de décision non-markoviens, elle pourrait s'avérer adaptée à notre problème. Elle ne donne malheureusement pas d'informations satisfaisantes sur la politique. En effet, si pour une observation o on a deux actions possibles a_1 et a_2 pour lesquelles les probabilités (localement) optimales trouvées $P(o,a_1)$ et $P(o,a_2)$ sont non nulles, alors leurs Q-valeurs respectives sont nécessairement égales : $Q^\pi(o,a_1) = Q^\pi(o,a_2)$, quelle que soit la différence entre ces probabilités (voir la description de l'algorithme en section 2.3.3.3). Ce rapide exemple montre que cette utilité n'est pas une mesure appropriée de l'importance relative de différentes décisions possibles.

Récompense décomptée

$$Q^\pi(o,a) \quad = \quad E^\pi \left[\sum_{k=1}^{\infty} \gamma^k r_k | o_0 = o, a_0 = a \right] \tag{6.7}$$

La plus classique espérance de gain décompté souffre de la perspective à court terme induite par son facteur de décompte γ ($\gamma \in [0,1)$). Elle reste pourtant la seule définition de Q-valeurs qui semble être une bonne référence pour pondérer les différentes actions. C'est la définition que nous avons décidé de conserver dans nos travaux.

Une petite remarque sur cette Q-valeur est qu'elle est habituellement rencontrée dans le Q-learning [Watkins, 1989], où la formule de mise à jour est spécifiquement adaptée à la recherche d'une politique *déterministe* optimale (d'où le max). Pour apprendre une estimation de l'espérance de gain décompté liée à l'accomplissement de l'action a quand l'agent observe o, on a employé le calcul plus approprié suivant :

$$Q(o,a) \quad \leftarrow \quad (1-\alpha) * Q(o,a) + \alpha * \left(r + \gamma \sum_{a' \in \mathcal{A}} [\pi(o',a') * Q(o',a')] \right) \tag{6.8}$$

où o' est la prochaine observation, r est le renforcement reçu et π la politique stochastique à évaluer.

Chacune des trois définitions de Q-valeurs vues ici permet une estimation à travers des expérimentations. Toutefois, une question qui n'est pas développée dans ces lignes est de comparer les complexités effectives des apprentissages de chacune d'elles.

On retiendra que c'est la troisième définition, **l'espérance de gain décompté**, qui sera utilisée par la suite, les autres apportant peu d'information utile.

6.2.3.2 A quoi servent les paramètres ?

Les Q-valeurs peuvent être de bonnes références pour comparer l'importance relative entre deux choix d'actions (notons qu'une probabilité nulle est aussi un choix) venant tous deux d'un même comportement : ils ont été appris précisément en tant que mesure d'une grandeur (l'espérance de gain) dans un espace observation-action. Même si la politique d'un comportement est utilisée dans un nouveau contexte (comme dans un comportement scalable), les Q-valeurs apprises restent une évaluation correcte d'une décision. En effet, si elles étaient ré-apprises dans ce nouveau contexte, toutes les Q-valeurs seraient généralement modulées d'une façon comparable : dans un monde plus grand par exemple, le principal changement est un rapprochement des valeurs vers zéro, puisque le temps avant d'obtenir une récompense est plus long.

Au contraire, les Q-valeurs de deux comportements de base distincts, si elles sont utilisées comme poids dans une nouvelle situation complexe, ne sont pas nécessairement directement comparables, selon la fonction de récompense de chacun d'eux ou selon la taille de l'environnement dans lequel elles ont été apprises. De plus, l'importance d'un comportement de base peut dépendre du problème dans lequel il est utilisé (tomber dans un trou pourrait être jugé plus dangereux dans un nouveau problème).

A cause de cela, un équilibrage est requis pour corriger ces phénomènes aussi bien que possible au sein de la combinaison. Nous proposons d'introduire un simple paramètre θ (pour chaque comportement de base), lequel servira à définir un facteur d'échelle $e^\theta > 0$ pour les Q-valeurs. L'utilisation de e^θ se justifie pour des raisons pratiques : cela permet de faire une exploration sur tout l'ensemble des réels alors que les échelles doivent être positives. La figure 6.4 illustre l'emploi de ce paramètre comme un biais dans la comparaison de deux valeurs.

6.2.3.3 Apprentissage

Ces paramètres θ qui viennent d'être décrits dépendent de la tâche complexe considérée, contrairement aux Q-valeurs. En tant que tels, ils doivent être appris spécifiquement, ce qui résulte en une combinaison adaptative des comportements de base.

Apprendre par l'intermédiaire d'une approche classique d'apprentissage par renforcement (telle que celles vues au chapitre 2) n'est pas possible dans la mesure où d'une part il ne s'agit pas d'apprendre directement une politique complète, et d'autre part un algorithme tel que la descente de gradient (en ligne) de Baxter et Bartlett [Baxter *et al.*, 2001] nécessite un certain

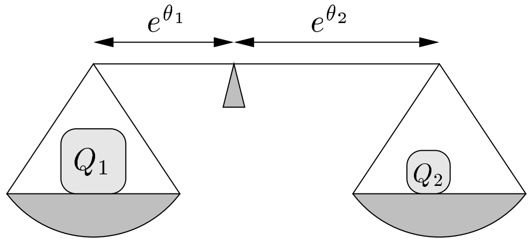


FIG. 6.4 – Les paramètres θ peuvent être vus comme définissant les longueurs des bras d'une balance, lesquels doivent être ajustés pour comparer Q_1 à $\frac{e^{\theta_2}}{e^{\theta_1}} * Q_2$.

nombre de propriétés mathématiques qui ne peuvent être ici assurées. Nous avons pour cette raison opté pour des algorithmes d'optimisation simples tels que le recuit simulé, la principale limitation étant que l'évaluation d'un ensemble donné de paramètres requiert une phase de simulation non négligeable.

L'apprentissage des paramètres θ est abordé de manière plus approfondie dans l'analyse de nos expériences. Pour l'instant nous allons nous concentrer sur les différentes fonctions de combinaison qui sont présentées dans ces expérimentations.

6.2.4 Formes spécifiques de f_{combi}

Ayant décrit une formulation générale hypothétique de f_{combi} et discuté des éléments (Q-valeurs et paramètres θ) qui vont peser sur la décision, nous pouvons maintenant entrer dans la présentation des formes spécifiques testées. Cette section débute par une combinaison particulière uniquement basée sur les Q-valeurs (elle servira de référence) et se continue avec des formulations additives puis multiplicatives inspirées par la formulation générale (équation 6.2), mais qui vont rapidement sortir de ce cadre.

6.2.4.1 Combinaison directe de Q-valeurs

La première forme est loin de ressembler à la formulation générale présentée auparavant. Elle est présentée pour la simple raison que les Q-valeurs sont souvent perçues comme directement liées à la politique suivie (en considérant la définition classique des Q-valeurs qui a été choisie). Cette combinaison est un calcul en deux étapes. D'abord, des Q-valeurs combinées sont obtenues pour chaque action a (et l'observation courante o) par

$$\mathcal{Q}_\pi(o,a) = \sum_{(b,c)\in\mathcal{BC}(o)} Q_b(o,c,a). \tag{6.9}$$

Ensuite, une distribution de probabilité sur les actions est calculée à travers une formule de type équation de Boltzmann :

$$\boxed{\mathcal{P}(o,a) = \frac{e^{\frac{\mathcal{Q}_\pi(o,a)}{T}}}{\sum_{b\in\mathcal{A}} e^{\frac{\mathcal{Q}_\pi(o,b)}{T}}}}, \tag{6.10}$$

où T est une température fixée, connue d'après de précédents essais pour donner des résultats satisfaisants sur de tels problèmes (c'est-à-dire des problèmes utilisant un Q-learning boltzmannien adapté pour obtenir une politique stochastique, comme décrit en section 2.3.3.1).

On retrouve ici presque la variante “maximiser la joie collective” du W-learning (formule 5.3). Mais pour montrer que le lien intuitif entre Q-valeurs et probabilités d'action peut être dangereux, un contre-exemple est développé en annexe en section A.2.

6.2.4.2 Un poids par configuration (+)

En considérant la formule générique de f_{combi} avec les opérateurs $+$ et $*$ (on parlera d'une combinaison *additive*, notée par le signe (+) dans le titre de la section), une première idée est d'employer l'utilité de l'observation d'une configuration $V_b(o,c)$ comme poids. Hélas cela pourrait donner des poids négatifs si seules des actions dangereuses sont envisageables (un comportement qui gère quatre trous entourant l'agent). De plus, aucun poids ne serait associé au comportement d'*évitement* d'un agent situé à côté d'un unique trou, comme le montre la formule développée ci-dessous. L'utilité serait nulle ($V_{b_e}(o,c)=0$) alors qu'il s'agit d'une distribution de probabilité importante signifiant de ne pas aller vers ce trou ($P_{b_e}(o,c,dir_{trou})=0$).

$$\begin{aligned}
\begin{bmatrix} \mathcal{P}(o,a_1) \\ \mathcal{P}(o,a_2) \\ \vdots \\ \mathcal{P}(o,a_n) \end{bmatrix} &= \frac{1}{K} * \left(e^{\theta_{b_e}} . V_{b_e}(o,c) * \begin{bmatrix} P_{b_e}(o,c,a_1) \\ P_{b_e}(o,c,a_2) \\ \vdots \\ P_{b_e}(o,c,a_n) \end{bmatrix} + \dots \right) \\
&= \frac{1}{K} * \left(e^{\theta_{b_e}} . 0 * \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \dots \right)
\end{aligned} \tag{6.11}$$

Comme nos comportements de base ne concerneront que des signaux de renforcement la plupart du temps nuls, nous proposons de remplacer $V_b(o,c) = \max_a Q_b(o,c,a)$ par $W_b(o,c) = \max_a |Q_b(o,c,a)|$, de façon à ce que les gains négatifs comme positifs puissent influencer les pondérations. Dans cette première formulation, on utilise $w(o,c,b,a) = e^{\theta_b}.W_b(o,c)$, ce qui donne :

$$\boxed{\mathcal{P}(o,a) = \frac{1}{K} \sum_{(b,c)\in\mathcal{BC}(o)} e^{\theta_b}.\max_{a'} |Q_b(o,c,a')| * P_b(o,c,a)} \tag{6.12}$$

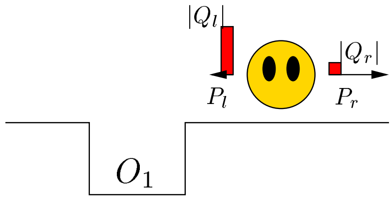
6.2.4.3 Un poids par configuration et par action (+)

Comme un poids unique pour toutes les actions peut sembler restrictif, une modification possible de la forme précédente est juste d'employer $w(o,b,c,a) = e^{\theta_b}.|Q_b(o,c,a)|$. Cela peut être illustré dans le même cas d'un agent côtoyant un trou, puisque cet agent n'a qu'à se préoccuper de la probabilité 0 dans la direction du trou. Si d'autres comportements ont de bons conseils à donner pour choisir parmi les autres directions, leurs voix seront prééminentes. La seconde formulation est ainsi :

$$\boxed{\mathcal{P}(o,a) = \frac{1}{K} \sum_{(b,c)\in\mathcal{BC}(o)} e^{\theta_b}.|Q_b(o,c,a)| * P_b(o,c,a)} \tag{6.13}$$

6.2.4.4 Un poids par configuration et par action (+) $\rightarrow$ correction

Un exemple simple (voir figure 6.5) prouve que la formule 6.13 précédente peut être largement améliorée. L'agent doit en l'occurrence choisir entre aller à `gauche` ou à `droite`, en essayant seulement d'*éviter* le trou O_1. Sur la figure ont été représentées les probabilités des actions (par des flèches) et les valeurs absolues des Q-valeurs (par des barres). La bonne action est ici d'aller à droite.



Une politique hypothétique π pourrait donner ici :

$$\begin{array}{ll} Q_l = -90 & Q_r = 1 \\ P_l = 0.1 & P_r = 0.9 \end{array}$$

FIG. 6.5 – Exemple justifiant la correction de la formule générale (voir texte).

En utilisant la formule de la section précédente, les probabilités obtenues sont (note : la dernière opération effectuée ci-dessous est la normalisation) :

$$\begin{aligned} \mathcal{P}_l &= \tfrac{1}{K} e^{\theta_{be}}.|Q_l| * P_l & \qquad \mathcal{P}_r &= \tfrac{1}{K} e^{\theta_{be}}.|Q_r| * P_r \\ &= \tfrac{1}{K} e^{\theta_{be}}.90 * 0.1 & &= \tfrac{1}{K} e^{\theta_{be}}.1 * 0.9 \\ &= \tfrac{1}{K} e^{\theta_{be}}.9 & &= \tfrac{1}{K} e^{\theta_{be}}.0.9 \\ &= \tfrac{9}{9.9} & &= \tfrac{0.9}{9.9} \end{aligned}$$

Alors que dans le comportement original l'agent va 10 fois plus souvent vers la droite que vers la gauche, le rapport est inversé dans la politique que l'on obtient au final.

L'erreur dans le processus de calcul est de ne faire qu'une normalisation finale (de manière à avoir une somme de probabilités égale à un). Nous suggérons de voir les comportements de base comme des "agents" donnant leurs avis d'abord pour obtenir une nouvelle probabilité moyenne pour chaque action (c'est-à-dire calculer la somme pondérée des probabilités pour chaque action indépendamment des autres) et ensuite seulement ces résultats sont normalisés pour assurer la propriété d'une distribution de probabilités précédemment citée.

La première étape est, pour toute action $a \in \mathcal{A}$, de calculer la probabilité estimée suivante :

$$\mathcal{R}(o,a) = \frac{1}{k(o,a)} \sum_{(b,c)\in\mathcal{BC}(o)} e^{\theta_b}.|Q_b(o,c,a)| * P_b(o,c,a) \tag{6.14}$$

$$\text{où } k(o,a) = \sum_{(b,c)\in\mathcal{BC}(o)} e^{\theta_b}.|Q_b(o,c,a)| \tag{6.15}$$

Les probabilités corrigées sont alors obtenues par :

$$\mathcal{P}(o,a) = \frac{1}{K}\mathcal{R}(o,a), \quad \text{où} \quad K = \sum_{a'\in\mathcal{A}} \mathcal{R}(o,a'). \tag{6.16}$$

On peut écrire cela à travers une unique formule plus synthétique (où les e^{θ_b} sont mis en commun) :

$$\boxed{\mathcal{P}(o,a) = \frac{1}{K}\frac{1}{k(o,a)} \sum_{b\in\mathcal{B}} \underbrace{e^{\theta_b}}_{\text{à apprendre}} \left[\underbrace{\sum_{c\in\mathcal{C}(o,b)} |Q_b(o,c,a)| * P_b(o,c,a)}_{\text{déjà connus}} \right]} \tag{6.17}$$

Une telle transformation pourrait aussi être accomplie sur d'autres formules. Nous ne la présentons que maintenant pour montrer comment les paramètres e^{θ_b} peuvent être factorisés.

Une remarque importante est que ce calcul amélioré ne respecte pas la forme générique que nous avons donnée (à cause du terme $\frac{1}{k(o,a)}$). Cette forme générique aura en fin de compte surtout servi de base pour en développer de plus spécifiques. Néanmoins, la modification amenée est efficace : son résultat sur l'exemple de la figure 6.5 est $\mathcal{P}_l = P_l$ et $\mathcal{P}_r = P_r$.

6.2.4.5 Un poids par configuration (∗)

Il n'y a pas de raison particulière d'utiliser des combinaisons arithmétiques (additives) plutôt que géométriques (multiplicatives) dans les heuristiques que nous développons (d'une façon similaire, [Tyrrell, 1993] fait un commentaire comparable sur un problème de combinaison de stimuli). C'est pourquoi nous avons aussi essayé les opérateurs $(*, x^y)$ (notés par le signe $(*)$ dans le titre de section).

La première formule multiplicative proposée n'utilise que les paramètres e^{θ} dans les poids (des expériences avec la fonction W définie auparavant n'ayant pas apporté de meilleurs résultats), d'où l'expression :

$$\boxed{\mathcal{P}(o,a) = \frac{1}{K} \prod_{(b,c)\in\mathcal{BC}(o)} P_b(o,c,a)^{e^{\theta_b}}} \tag{6.18}$$

Stabilité

Comme nous travaillons sur des combinaisons multiplicatives, un problème de stabilité apparaît. En effet, s'il y a un facteur nul dans chaque produit résultant en une probabilité d'action ($\forall a \in \mathcal{A}$, $\prod_{(b,c)\in\mathcal{BC}(o)} P_b(o,c,a)^{e^{\theta_b}} = 0$), on va tomber sur un calcul problématique : K étant lui-même nul, on obtient une division par zéro.

Apparaît en fait ici la nécessité que l'opération que constitue la combinaison de comportements respecte certaines propriétés :

- *stabilité* : qu'elle fournisse une distribution de probabilités, et
- *commutativité* : que l'ordre dans lequel sont traîtés les comportements soit indifférent.

Dans les différentes formes de combinaison présentées, la commutativité ne pose pas de problème. Par contre, dans le cas présent, il faut "interdire" les probabilités nulles dans les politiques des comportements de base pour assurer la stabilité. Cette nécessité impose de légères modifications des politiques (remplacer tout 0 par $\epsilon > 0$, puis renormaliser), puisque la plupart vont souffrir de ce "défaut".

6.2.4.6 Un poids par configuration et par action (∗) → correction

Ce dernier processus de combinaison est la contrepartie multiplicative de la combinaison présentée en section 6.2.4.4.

$$\mathcal{R}(o,a) = \left(\prod_{(b,c)\in\mathcal{BC}(o)} [P_b(o,c,a)]^{e^{\theta_b}.|Q_b(o,c,a)|} \right)^{\frac{1}{k(o,a)}} \tag{6.19}$$

$$\text{où } k(o,a) = \sum_{(b,c)\in\mathcal{BC}(o)} e^{\theta_b}.|Q_b(o,c,a)| \tag{6.20}$$

Pour illustrer la relation avec la version additive, notons que cette première équation est équivalente à la suivante :

$$\ln(\mathcal{R}(o,a)) = \frac{1}{k(o,a)} \sum_{(b,c)\in\mathcal{BC}(o)} e^{\theta_b}.|Q_b(o,c,a)| * ln(P_b(o,c,a)) \tag{6.21}$$

La seconde étape (normalisation des probabilités) reste la même et donne :

$$\boxed{\mathcal{P}(o,a) = \frac{1}{K}\left(\prod_{(b,c)\in\mathcal{BC}(o)} [P_b(o,c,a)]^{e^{\theta_b}.|Q_b(o,c,a)|}\right)^{\frac{1}{k(o,a)}}} \tag{6.22}$$

Résumé

Comme cette présentation a été plutôt longue, la table 6.1 fait un rapide rappel de la liste des méthodes de combinaisons exposées, de façon à les avoir toutes en tête (note : leurs numéros seront désormais utilisés pour les identifier).

6.3 Application/Expériences

Les expériences menées l'ont été sur le problème du monde des tuiles déjà présenté dans le chapitre précédent en section 5.3.2. Nous ne reviendrons donc pas sur sa description.

6.3.1 Les diverses combinaisons

6.3.1.1 Expériences effectuées

Pour évaluer l'efficacité de notre algorithme avec les différentes formes de combinaisons présentées en section 6.2.4, il a été testé dans différents environnements : la taille de la grille a été fixée à 6×6, mais le nombre d'objets (tuiles et trous) changeant.

Dans l'idéal, la meilleure politique stochastique devrait être utilisée comme référence dans chacun de ces cas. Malheureusement, la seule façon de l'obtenir est de passer par un apprentissage par renforcement. Une descente de gradient ne réussissant pas ici à apprendre plus qu'à éviter les trous (si ce n'est avec une tuile et un trou), la meilleure référence disponible a été obtenue avec un Q-learning boltzmannien adapté, lequel doit souvent être loin de l'optimum, mais montre les résultats qu'une méthode classique peut amener.

La table 6.2 présente dans sa première colonne les situations dans lesquelles des tests ont été conduits. Les résultats sont montrés dans la deuxième colonne[46], où les barres indiquent la récompense moyenne obtenue après une phase d'apprentissage des deux paramètres (un par comportement), et les barres d'erreur donnent l'écart-type observé. Dans chaque cas, 10 simulations ont été effectuées avec :

- une phase d'apprentissage de $70 * 100\,000$ pas de temps (en utilisant un recuit simulé), et
- une phase d'évaluation de $10 * 100\,000$ pas de temps.

Cette "unité de base" de 100 000 pas de temps est justifiée pour évaluer de manière précise la qualité d'un ensemble de paramètres. Finalement, une ligne horizontale est dessinée dans chaque figure au niveau atteint avec le Q-learning boltzmannien adapté (la température $T_\infty = 0,08$ utilisée a été choisie à travers diverses expérimentations sur nos problèmes-jeu).

[46] La troisième colonne sera utilisée plus tard.

TAB. 6.1 – Récapitulation des méthodes de combinaison proposées.

1. Combinaison directe de Q-valeurs

$$\mathcal{P}(o,a) = \frac{e^{\frac{\mathcal{Q}_\pi(o,a)}{T}}}{\sum_{b \in \mathcal{A}} e^{\frac{\mathcal{Q}_\pi(o,b)}{T}}}$$

2. (+) Un poids par configuration

$$\mathcal{P}(o,a) = \frac{1}{K} \sum_{(b,c) \in \mathcal{BC}(o)} e^{\theta_b} . \max_{a'} |Q_b(o,c,a')| * P_b(o,c,a)$$

3. (+) Un poids par configuration et par action

$$\mathcal{P}(o,a) = \frac{1}{K} \sum_{(b,c) \in \mathcal{BC}(o)} e^{\theta_b} . |Q_b(o,c,a)| * P_b(o,c,a)$$

4. (+) Un poids par configuration et par action → Correction

$$\mathcal{P}(o,a) = \frac{1}{K} \frac{1}{k(o,a)} \sum_{b \in \mathcal{B}} e^{\theta_b} \left[\sum_{c \in \mathcal{C}(o,b)} |Q_b(o,c,a)| * P_b(o,c,a) \right]$$

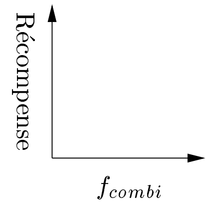
5. (∗) Un poids par configuration

$$\mathcal{P}(o,a) = \frac{1}{K} \prod_{(b,c) \in \mathcal{BC}(o)} P_b(o,c,a)^{e^{\theta_b}}$$

6. (∗) Un poids par configuration et par action → Correction

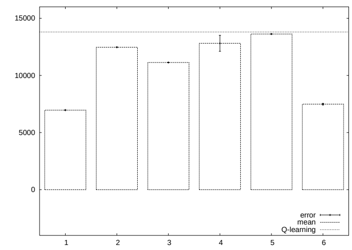
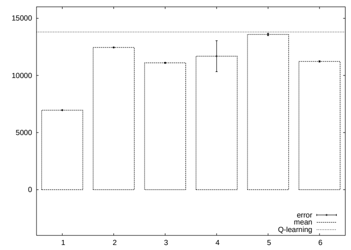
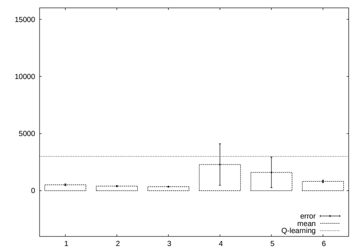
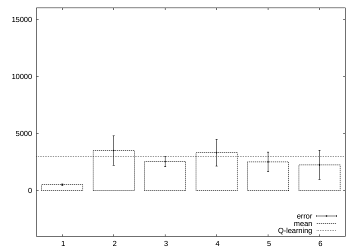
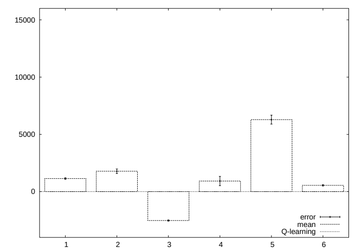
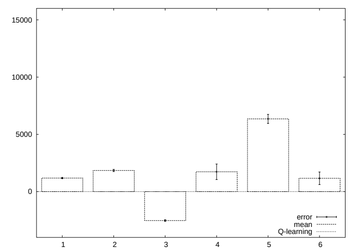
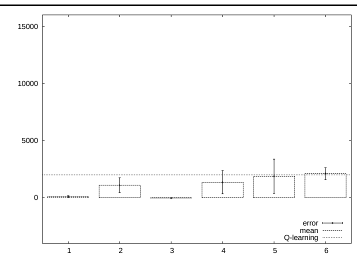
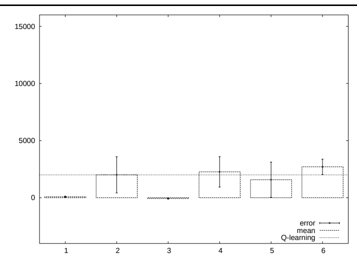
$$\mathcal{P}(o,a) = \frac{1}{K} \left(\prod_{(b,c) \in \mathcal{BC}(o)} [P_b(o,c,a)]^{e^{\theta_b} . |Q_b(o,c,a)|} \right)^{\frac{1}{k(o,a)}}$$

TAB. 6.2 – Efficacité des différentes combinaisons.



Dans chaque situation, on donne la récompense moyenne atteinte dans nos expérimentations pour 100 000 pas de temps, ainsi que l'écart-type. Les figures dans la dernière colonne montrent les mêmes tests avec l'utilisation d'un comportement *aléatoire* additionnel (voir section 6.3.2).

Sur les figures suivantes, l'axe des abscisses indique les combinaisons utilisées f_{combi}, et l'axe des ordonnées est consacré à la mesure de récompense.

$\#tuiles + trous$	combinaisons normales	...avec un comportement aléatoire
1 + 1		
2 + 1		
1 + 2		
2 + 2		

1. Combinaison directe de Q-valeurs
2. (+) Un poids par configuration
3. (+) Un poids par configuration et action
4. (+) Un poids par configuration et action → correction
5. (∗) Un poids par configuration
6. (∗) Un poids par configuration et action → correction

6.3.1.2 Analyse des résultats

Première impression. Une première remarque est que, globalement, les combinaisons semblent être toutes très efficaces avec une tuile et un trou, et subir une dégradation notable dans les autres cas. Toutefois, le Q-learning suit une évolution similaire.

Situations bloquantes. L'observation directe des comportements scalables de l'agent en activité amène des explications sur les problèmes rencontrés avec plus d'une tuile et d'un trou. En effet, deux situations de blocage typiques apparaissent (montrées sur les figures 6.6 et 6.7) dans lesquelles l'agent passe la plupart de son temps.

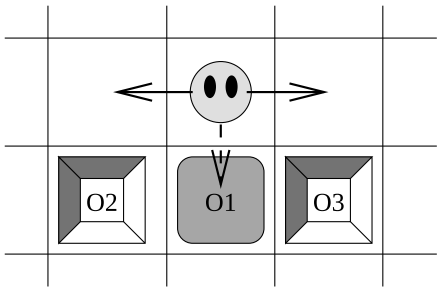


Fig. 6.6 – Blocage avec 1 tuile et 2 trous.

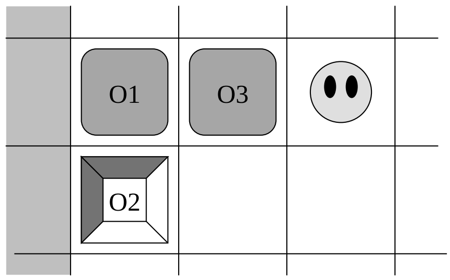


Fig. 6.7 – Blocage avec 2 tuiles et 1 trou (et des cases interdites à l'ouest).

Note : Les décisions inappropriées sont non seulement celles prises dans ces situations, mais aussi celles qui y mènent.

Dans les deux cas, l'action appropriée n'est suggérée par aucune des paires (bb, cfg) impliquées. Dans la figure 6.6, l'agent hésite entre aller vers l'est ou vers l'ouest (d'où un mouvement oscillant), et dans la figure 6.7, les deux instances du comportement `pousser` veulent que l'agent aille vers l'ouest (alors que cette direction est bloquée par un mur). Ainsi, de tels phénomènes amènent des limitations fortes au principe de combiner des comportements élémentaires.

Des combinaisons vraiment mauvaises. On peut toutefois faire d'autres commentaires intéressants grâce à ces tests. Par exemple, l'idée qu'"estimer une politique d'après les Q-valeurs n'est pas une bonne intuition" est confirmée par les mauvais résultats de la combinaison 1 (combinaison de Q-valeurs). Il est aussi confirmé que la combinaison 3 ((+) un poids par configuration et action, *non* corrigé) ne devrait pas être utilisée, puisqu'elle amène même à produire un comportement scalable avec une récompense moyenne négative (dans le cas 1-tuile/2-trous).

Des résultats instables. Quand il y a deux tuiles dans l'environnement, la plupart des approches efficaces semblent donner des résultats très instables. Pendant l'exploration, le système est enclin à tomber régulièrement dans des optima locaux. Cela pourrait probablement être amélioré par des réglages du recuit simulé, mais aux dépens du temps de calcul (lequel est déjà important).

Des poids dépendant de l'action ? La combinaison 4 ((+) un poids par configuration et action, corrigé) peut être vue comme une évolution de la combinaison 2 ((+) un poids par configuration) vers des pondérations dépendant de l'action. Néanmoins, la comparaison des deux algorithmes additifs ne prouve pas que des poids dépendant de l'action constituent une amélioration. En comparant les combinaisons 5 et 6 (dans le cas 1-tuile/2-trous), des poids ne dépendant que du comportement peuvent apparaître comme la meilleure solution. Il serait toutefois hasardeux de tirer des conclusions des présents résultats.

6.3.2 Un comportement aléatoire ?

6.3.2.1 Expériences effectuées

Une idée courante pour sortir un agent des situations bloquantes est d'ajouter du bruit à son système de prise de décision (ici sa politique). Jusqu'à un certain point, un tel bruit est bénéfique. Au delà de ce point, il sera la cause d'une dégradation progressive de l'efficacité de l'agent.

Pour mettre ce principe en application et aider à résoudre les deux blocages présentés dans les figures 6.6 et 6.7, un nouveau comportement de base artificiel a été créé. Ce comportement "aléatoire" est défini comme suit :

1. $\mathcal{C}^T_{b_e}$ est vide (il n'y a pas d'objets dont il faille se préoccuper).
2. $P_{b_e}(o, c, a)$ est équiprobable (toutes les actions ont toujours des chances égales d'être choisies).
3. $Q_{b_e}(o, c, a)$ est constante et non-nulle (sinon son poids serait nul dans toutes les combinaisons, et ce comportement ne servirait à rien).

La phase d'adaptation des paramètres servira à régler le volume du bruit nécessaire. Revenant à la table 6.2, les quatre figures dans la troisième colonne correspondent aux mêmes expérimentations que leurs voisines, mais en faisant usage de comportement de base aléatoire additionnel.

6.3.2.2 Analyse des résultats

En regardant le cas "non-pathologique" à 1 tuile et 1 trou, le comportement aléatoire ne semble pas améliorer les résultats obtenus. Au contraire, une légère baisse peut être observée. Ce point est à relier au plus grand nombre de paramètres à ajuster, lequel agrandit l'espace à explorer.

Dans les trois autres cas de cette étude, la plupart des combinaisons bénéficient de ce nouveau comportement de base. C'est évident dans les situations impliquant 2 tuiles et au moins 1 trou, probablement du fait d'une plus grande efficacité pour le problème décrit par la figure 6.7. Si cela ne résoud pas complètement les blocages, cela peut aider dans le cas de combinaisons "trop déterministes".

6.3.3 Réutilisabilité et "scalabilité"

L'intérêt d'une méthode de sélection d'action ne réside pas uniquement dans sa capacité à résoudre un problème donné. On va donc voir ici si notre approche permet de réutiliser les paramètres appris, c'est-à-dire voir dans quelle mesure elle est "scalable".

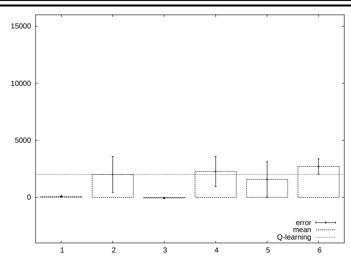
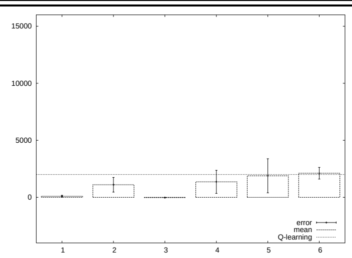
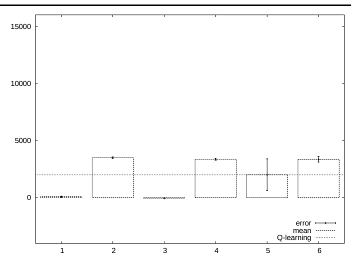
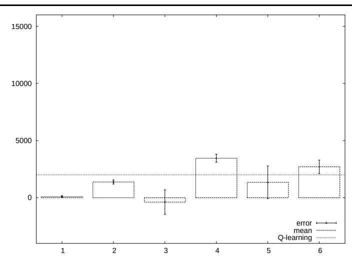
6.3.3.1 Expériences effectuées

Pour l'instant, les tests se sont concentrés sur l'efficacité de la combinaison de comportements de base. Un point qui n'a pas encore été pris en compte est la possibilité de réutiliser des paramètres d'une situation à l'autre. C'est le sujet de ces dernières expériences, qui s'organisent en deux parties :

1. Dans la table 6.3 est comparé l'apprentissage de paramètres dans le cas 2-tuiles/2-trous (désormais noté $(2+2)$) quand ils sont initialisés soit avec des valeurs nulles, soit avec la valeur moyenne des meilleurs paramètres appris dans les cas $(1+2)$ et $(2+1)$.

2. Puis, la table 6.4 montre l'efficacité des différentes combinaisons quand on réutilise les meilleurs paramètres pour $(2+2)$ dans des situations mettant en jeu un plus grand nombre d'objets.

TAB. 6.3 – L'apprentissage est-il meilleur quand on réutilise des paramètres ?

<table>
<tr><th>Initialisation</th><th>combinaisons normales</th><th>...avec comportement aléatoire</th></tr>
<tr><td>en partant de zéro</td><td>
</td><td>
</td></tr>
<tr><td>en réutilisant des paramètres</td><td>
</td><td>
</td></tr>
</table>

6.3.3.2 Analyse des résultats

La première table de figures montre clairement l'intérêt qu'on a à ne pas commencer l'adaptation des paramètres en partant de zéro. Les paramètres de départ calculés donnent de bonnes initialisations : les niveaux moyens d'efficacité obtenus sont notablement meilleurs et plus stables. Même s'il peut être nécessaire d'apprendre avec précaution les paramètres dans les deux cas simples ($(1+2)$ et $(2+1)$), c'est un travail effectué pour le long terme.

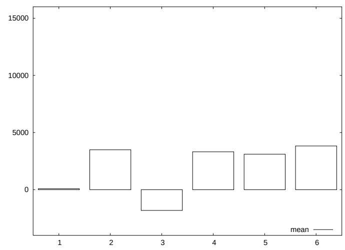
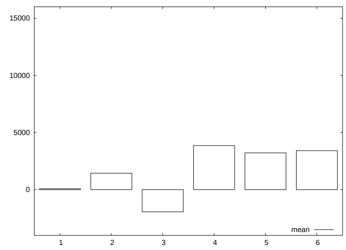
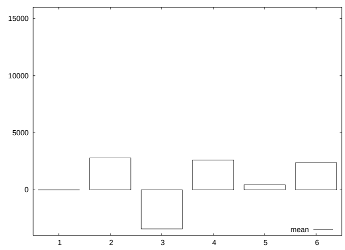
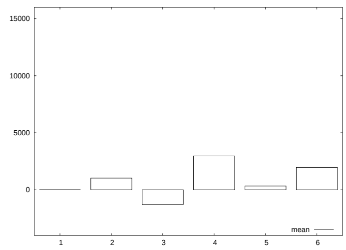
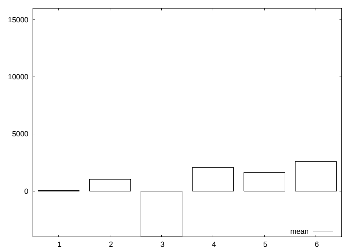
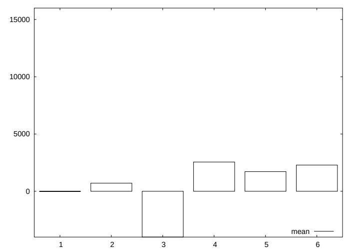
Comme l'illustre la dernière table, quelques combinaisons semblent mieux "passer à l'échelle", puisqu'une simple réutilisation de bons paramètres du cas $(2+2)$ amène à des résultats très stables et très satisfaisants. C'est particulièrement remarquable dans les approches 4 et 6, ce qui fournit un argument en faveur de ces deux méthodes dans lesquelles les poids dépendent aussi de l'action. Ces algorithmes computationnellement plus complexes peuvent s'avérer intéressants avec un nombre croissant de comportements à combiner. D'autres exemples que le monde des tuiles devront toutefois être expérimentés pour renforcer cette hypothèse.

6.4 Conclusion

Après avoir discuté des approches existantes dans le domaine de la sélection d'action et de la décomposition de PDM (chapitre 5), le présent chapitre a proposé une *architecture de sélection d'action* se basant sur des comportements de base appris par renforcement, et répondant aux souhaits que nous avions formulés :

– que ce soit une *hiérarchie de type flux libre* (de façon à trouver un compromis parmi les comportements concurrents),
– que le *comportement produit soit stochastique* (ce qui évite dans une certaine mesure de provoquer des blocages), et enfin

TAB. 6.4 – A quel point peut-on réutiliser des paramètres ?

$\#tuiles + trous$	combinaisons normales	...avec comportement aléatoire
2 + 2		
3 + 2		
2 + 3		

– que les *paramètres de la combinaison soient, dans la mesure du possible, réglés automatiquement* (en faisant appel à des outils classiques d'optimisation).

Les travaux effectués ont dans l'ensemble répondu à ces attentes, même s'il ne s'agit que d'heuristiques, et qu'on peut toujours se demander si une autre forme de combinaison ne serait pas meilleure. Ils ont aussi soulevé quelques difficultés persistantes ou aspects intéressants, parmi lesquels on peut citer :

– la réutilisabilité des paramètres θ appris : ils peuvent efficacement servir de base dans des situations plus complexes que celles auxquelles ils étaient dédiés ;
– l'intérêt, même s'il reste limité, d'ajouter un comportement de base "aléatoire" qui permet d'ajouter du bruit, et de sortir de blocages persistants ; et
– ces mêmes blocages qui font penser que, si un comportement combiné peut être très loin des performances optimales, il est probablement assez proche d'un bon comportement (à ces situations de blocages près).

De cette dernière remarque est venue l'idée de faire un apprentissage de politique sur la base d'une politique combinée. C'est cette idée que le chapitre 7 va aborder.

Chapitre 7

Apprentissage incrémental sur la base d'une combinaison

Sommaire

Au cours du chapitre précédent a été menée une étude sur notre approche de combinaison de comportements. Cette étude a révélé que, pour les formes de combinaisons efficaces trouvées, la politique obtenue est généralement "proche d'une bonne politique" : la grande majorité des actions proposées sont de bonnes décisions, et seules quelques situations sont mal traîtées, provoquant de coûteux blocages (des culs-de-sac dont l'agent ne sait sortir).

Cette constatation nous a conduit à l'idée que, comme une politique ainsi produite "ressemble" a une bonne solution, elle pourrait efficacement faire office d'*initialisation* pour une recherche directe de politique (un apprentissage complet, et non plus une simple combinaison). Cela permettrait même de calculer un nouveau *comportement simple* (voir définition 8 en page 143), dans la perspective de s'en servir éventuellement comme nouveau *comportement de base*.

En suivant cette idée, ce court chapitre va, en section 7.1, appeler l'attention de manière approfondie sur les raisons de ce nouveau développement, montrer dans la section 7.2 le principe sur lequel il est basé, et finalement discuter (section 7.3) de son efficacité en pratique sur le banc d'essai qu'est pour nous le monde des tuiles.

Note : Le travail présenté dans ce chapitre (comme dans le chapitre suivant d'ailleurs) ne s'applique pas de manière exclusive à l'une ou l'autre combinaison étudiée au chapitre précédent[47]. Ainsi, on se limitera à une étude effectuée dans le cadre de la combinaison 4 (voir table 6.1). Si ce n'est pas de manière évidente la meilleure combinaison, elle reste une des plus satisfaisante (et des plus naturelles, du fait de son caractère additif).

7.1 Motivation

La raison principale ayant motivé un nouvel algorithme est la nécessité de dépasser des situations que l'on peut qualifier de "non-linéaires". Comme on l'a expliqué en section 6.3.1.2 (et comme illustré sur les figures 6.6 et 6.7) certaines configurations ne peuvent trouver de solution par l'intermédiaire d'une combinaison linéaire de comportements de base. Ces configurations requièrent la prise en compte *simultanée* d'un plus grand nombre d'éléments parmi ceux présents dans la perception à gérer.

Pour répondre à cette difficulté, deux solutions ont été considérées :

7.1.1 Apprendre des règles d'"exception"

Une première idée est de se contenter d'apprendre des règles spécifiques pour ne corriger que les situations critiques, comme le proposent [Gadanho et Custodio, 2002]. Malheureusement, pour une situation problématique donnée, de nombreuses règles vont devoir être apprises pour *aussi* traîter une partie des situations précédentes : celles qui mènent à la situation de blocage. En effet, si à partir des situations $\sigma_1, \dots \sigma_n$ l'agent va à la situation bloquante σ_0 (voir figure 7.1-a), et si changer le comportement face à σ_0 renvoie vers une des situations parmi $\sigma_1, \dots \sigma_n$ (comme c'est le cas en figure 7.1-b), alors on va obtenir un comportement oscillatoire (on pourrait parler de "blocage de degré 2"), qui ne fait que déplacer le problème[48].

En outre, il ne serait pas évident de combiner de telles règles avec d'autres comportements, et l'on ne saurait comment régler des conflits entre règles simultanément applicables. Notre objectif n'est pas d'avoir une solution appliquable seulement momentanément pour *un* cadre de travail fixe, mais qu'elle puisse être réutilisée dans d'autres cas. Ainsi nous avons préféré opter pour une seconde solution, que nous présentons maintenant.

[47]Exception faite de la combinaison directe de Q-valeurs, cas très particulier de notre étude.

[48]On peut en fait dès le départ avoir des blocages d'un quelconque degré.

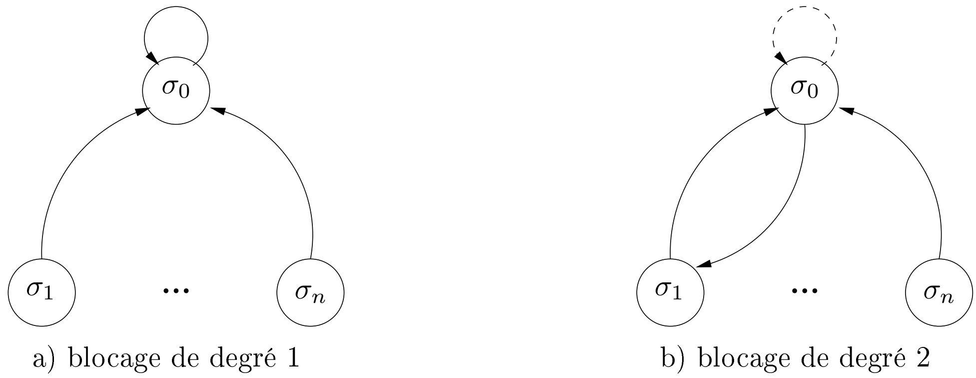


FIG. 7.1 – Ce qui se passe quand on se contente d'ajouter *une* règle pour lever un blocage simple (ici dans un cas déterministe).

7.1.2 Apprendre de nouveaux comportements de base

Une autre solution est d'apprendre de nouveaux comportements de base complets pour chaque type de configuration complexe qui ne serait pas correctement géré par la combinaison des comportements de base initialement prévus. Dans le cas de la figure 6.7, il s'agirait donc simplement d'apprendre un comportement simple impliquant deux tuiles et un trou.

C'est un solution certes rudimentaire, mais là où une règle d'"exception" apprise dans un cadre fixé ne peut être réutilisée de manière systématique dans de nouvelles situations, un *nouveau comportement* peut pour sa part venir simplement s'ajouter aux comportements de base déjà utilisés. Ce nouveau comportement de base vient ainsi proposer ses décisions là où les autres ne savaient auparavant comment agir.

Approche incrémentale suivie

Malheureusement, on a vu dans les expérimentations présentées en section 6.3.1.1 qu'apprendre un comportement complet s'avère particulièrement difficile pour un agent néophyte, même avec un nombre d'observations possibles apparemment raisonnable. C'est précisément ce qui arrive si l'agent essaye d'apprendre de manière directe un comportement "pousser" (une tuile dans un trou) avec deux tuiles et un trou comme il le faut dans le cas de la figure 6.7.

Or on a aussi déjà vu qu'un comportement obtenu par combinaison b_c est proche d'un bon comportement (seules quelques observations posant problème). L'idée que nous proposons de suivre est alors de faciliter l'apprentissage de chaque nouveau comportement b_n en le faisant débuter à partir du comportement combiné b_c (dont seuls les paramètres θ ont dû être appris), lequel est amélioré à travers un apprentissage par montée de gradient.

7.2 Description

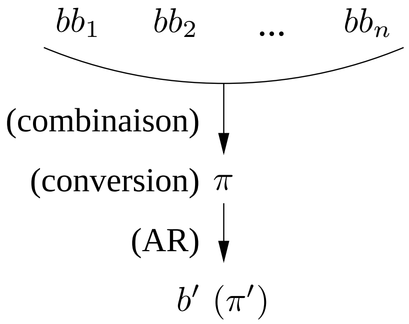
Nous présentons maintenant le schéma général de l'approche incrémentale proposée, laquelle vise donc à concevoir de nouveaux comportements simples, pouvant éventuellement servir de comportements de base.

7.2.1 Principe

Pour définir le nouveau comportement à apprendre, il faut expliciter son type de configuration $\mathcal{C}^T$ et une récompense globale $\mathcal{R}$ qui fera office de motivation. Les trois phases consécutives qui suivent forment alors l'algorithme employé, comme l'illustre aussi la figure 7.2 :

1. *Combinaison :* Un ensemble de comportements de base $BB = \{bb_1, bb_2 \dots bb_n\}$ étant choisi et un nouveau "but" étant considéré (décrit par la fonction $\mathcal{R}$), l'algorithme 5 (page 142) est exécuté pour adapter une combinaison de comportements à travers la recherche de paramètres de pondération optimaux.
2. *Conversion :* Le *comportement scalable* obtenu (voir définition 10 page 144) par cette combinaison optimale est converti en un comportement d'initialisation $b = \langle \mathcal{C}^T, \pi, Q \rangle$.

   Un problème est de pouvoir réutiliser ces données pour faire un nouvel apprentissage. Dans notre cas, la politique π est décrite comme une politique stochastique paramétrée propre à l'utilisation de l'un des algorithmes de montée de gradient de Baxter et Bartlett [Baxter et Bartlett, 2001; Baxter *et al.*, 2001], avec la paramétrisation décrite en section 2.3.3.2. La table des Q-valeurs, elle, est initialement mise à zéro.
3. *Apprentissage :* π sert de politique de départ pour apprendre un comportement complet b' (de politique localement optimale π') par l'algorithme en ligne OLPOMDP que nous avons déjà présenté (toujours en section 2.3.3.2). On en profite pour estimer simultanément les Q-valeurs (sur l'espace observation-action) associées à cette politique.



L'algorithme proposé combine les comportements de base bb_1, $bb_2 \dots bb_n$ (apprend les paramètres de pondération), obtient une politique π, et l'utilise comme racine d'une nouvelle phase d'apprentissage conduisant à un nouveau comportement b' (de politique localement optimale π').

FIG. 7.2 – Principe de la méthode incrémentale.

Pour compléter cette description générale de l'algorithme, nous en développons quelques aspects ci-après.

7.2.2 Compléments

7.2.2.1 Estimation des Q-valeurs

Comme on l'a noté, du fait que l'on cherche à obtenir un nouveau *comportement simple* (dans l'idée de l'utiliser comme comportement de base), l'un des objectifs de l'algorithme qui vient d'être présenté est aussi d'apprendre une estimation de la table des Q-valeurs correspondant à π' (politique optimale obtenue).

Pour rappel, en étudiant dans le chapitre précédent les formes de poids à utiliser, on a choisi de faire usage, en tant que Q-valeurs, de la classique espérance de gain décomptée (les autres définitions considérées s'avérant inappropriées à l'usage que nous souhaitions en faire). Avec ce choix, et en prenant en compte le caractère non-markovien de notre cadre de travail, la formule de mise à jour devient simplement (c'est l'équation 6.8) :

$$Q(o,a) \quad \leftarrow \quad (1-\alpha) * Q(o,a) + \alpha * \left(r + \gamma \sum_{a' \in \mathcal{A}} [\pi(o',a') * Q(o',a')] \right)$$

Ayant déjà abordé ce sujet (page 147), nous n'entrerons pas ici dans plus de détails. Ce bref rappel devrait être suffisant.

Par contre, on va pouvoir s'intéresser à la définition de la fonction de récompense que va requérir la dernière phase de l'algorithme proposé, celle de l'apprentissage.

7.2.2.2 Et en ce qui concerne les récompenses ?

Un autre aspect dont il faut discuter est la façon de gérer les récompenses lors de la conception de nouveaux comportements simples.

Le problème du monde des tuiles considéré dans nos expérimentations ne correspond pas à une situation où seule une forme unique de récompense est accessible. Les deux comportements de base donnés en exemple correspondent à deux causes différentes de récompenses : l'une négative en cas de chute dans un trou, et l'autre positive si une tuile est poussée dans un trou. C'est même sur cette idée de *plusieurs* sources de signaux de renforcement qu'est fondée notre idée de combinaison de comportements.

On va donc regarder maintenant les possibilités offertes par la différenciation des sources de récompenses, et discuter en même temps des limites de la décomposition ainsi opérée.

Récompenses élémentaires

Lors de la réalisation d'un agent, être capable de distinguer les sources de renforcements semble une hypothèse raisonnable. Cela dépend toutefois du point de vue adopté dans la conception de l'agent. Mais dans les cas appropriés, nous suggérons d'introduire à dessein ces différentes sources de renforcement sous la forme de *récompenses élémentaires* (notées $\mathcal{R}^e$). Techniquement, cela rend possible la sélection des buts suivis par l'agent au cours de l'apprentissage d'un nouveau comportement.

Cette capacité de séparer les comportements de base selon des types de récompense a un double intérêt :

- D'une part cela permet d'affiner la "classification" de ces comportements de base, et donc éventuellement de les utiliser de diverses façons selon qu'ils agissent comme des motivations inhibitrices ou incitatrices (par exemple).
- D'autre part, les algorithmes d'apprentissage par renforcement sont enclins à tomber dans des optima locaux. Cela va pouvoir être évité en faisant des apprentissages préliminaires qui ne tiennent compte que d'un but parmi plusieurs.

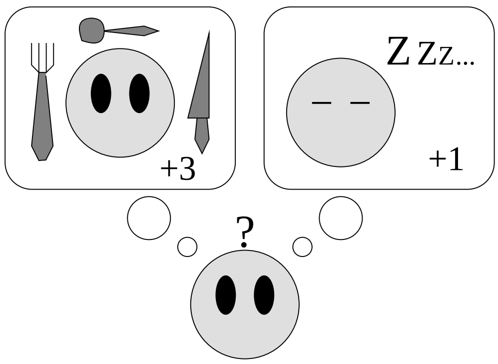
Le premier argument concerne des perspectives de recherche de meilleures formes de combinaisons (tenant compte de cette possible classification). Mais il n'a pas été mis en œuvre au cours de cette thèse. Au contraire, le second argument va apporter des améliorations directes à la méthodologie employée, comme le reflèteront nos expérimentations sur le monde des tuiles.

Combinaison de récompenses

Un quelconque comportement pouvant mettre en jeu plusieurs sources de renforcement, on parlera assez naturellement de *combinaison de récompenses* $\mathcal{R}^c$ pour désigner la fonction résultant de l'ensemble des récompenses élémentaires mises en jeu. On considère alors classiquement qu'on a une relation additive entre les signaux élémentaires, ce qui s'écrit :

$$\mathcal{R}^c = \sum_i \mathcal{R}^{e_i} \tag{7.1}$$

Cette discussion nous ramène au problème du soin qui doit être apporté à la définition des fonctions de récompense (déjà abordé en section 1.3.1.1). Dans notre présent problème de combinaison de récompenses, il faut bien noter que définir des fonctions de récompense élémentaires, lesquelles permettent d'obtenir des politiques satisfaisant séparément chacun des objectifs visés, n'est pas suffisant. L'interaction entre buts peut rendre la re-combinaison problématique, comme illustré par l'exemple de la figure 7.3. Mais la principale question qui va se poser dans notre cas est de savoir comment pondérer les différents gains en jeu (nous faisons l'hypothèse de récompenses indépendantes).



Supposons un agent doté de deux récompenses élémentaires : l'une quand il `mange` et l'autre quand il `dort`. Toute combinaison simple de ces deux récompenses va conduire l'agent à ne faire que manger, puisque c'est l'activité la plus rémunératrice.
Assez logiquement, le modèle de renforcement choisi est mal choisi, et c'est un compromis entre les deux activités qui serait le plus sain. Une solution possible est ici de faire intervenir une notion de *ressources* à gérer.

FIG. 7.3 – Pourquoi décomposer en récompenses élémentaires n'est pas simple.

Bilan

En résumé, décomposer un signal de renforcement en signaux élémentaires va permettre de distinguer plus finement les tâches pour lesquelles un agent doit connaître un comportement de base. On simplifie d'ailleurs d'autant les apprentissages en s'attachant à des objectifs plus simples. Le problème est qu'une telle décomposition n'est pas toujours viable : si deux buts sont liés, on risque d'avoir des difficultés ne serait-ce qu'au moment de recomposer les comportements obtenus.

Note : Mentionnons finalement le fait que tenir compte de types de récompense était sans utilité quand il ne s'agissait que de combiner des comportements de base, puisque les politiques et les Q-valeurs suffisaient à résumer l'information utile. C'est en fait une question qui est apparue quand il s'est agi de gérer l'initialisation de nouveaux comportements simples.

7.3 Expériences

Pour évaluer la méthode décrite dans ce chapitre, nous nous sommes ici encore appuyés sur des expériences conduites dans le monde des tuiles. Après avoir précisé la méthodologie appliquée, nous exhiberons et commenterons les résultats de nos simulations.

7.3.1 Méthodologie

Algorithmes comparés

Pour cet algorithme, trois façons d'obtenir des politiques dédiées à un problème donné ont été testées :

1. *Table rase* : apprend la politique à partir de zéro à travers une classique recherche directe de politique (dans l'espace des politiques stochastiques) : la montée de gradient de Baxter (section 2.3.3.2). Cette première méthode est la référence, même si elle peut ne pas être couronnée de succès (à cause de difficultés de convergence, souvent dues à des optima locaux).
2. *Par combinaison* : apprend les paramètres d'une combinaison de comportements de base telle que décrite au chapitre 6 (comme on l'a dit, il s'agit plus précisément ici de la combinaison 4).
3. *Incrémentalement* : utilise l'approche incrémentale que nous étudions justement dans ce chapitre.

Ces deux dernières méthodes correspondent aux deux étapes principales de notre algorithme, puisque la deuxième méthode aboutit à un résultat intermédiaire dans le déroulement de la troisième. Les comparer permettra de mesurer l'intérêt de la phase d'apprentissage par renforcement qui a été ajoutée.

De plus, cette fois-ci, quand un apprentissage de politique est effectué (méthodes 1 et 3), c'est à chaque fois l'algorithme OLPOMDP qui est utiltisé. Même si la référence (table rase) va amener d'assez mauvais résultats, cela permettra une comparaison directe avec notre méthode.

Dans la table 7.1 qui liste les résultats de nos expériences, les trois dernières colonnes sont dédiées respectivement à ces trois différentes méthodes. Mais nous commenterons ces résultats une fois la méthodologie appliquée décrite de manière complète.

Conditions de simulation

La table 7.1 présente dans ses deux premières colonnes les conditions de simulation qui ont été suivies. Dans une colonne apparaît le nombre de tuiles et de trous dans les simulations effectuées, et dans l'autre est indiqué la combinaison de récompenses impliquée. Pour ce qui est de cette combinaison de récompenses, on rencontre les trois cas suivants :

- une récompense positive, quand une tuile est poussée dans un trou, symbolisée par un "+",
- une récompense négative, quand l'agent passe dans un trou, symbolisée par un "−", et
- quand les deux récompenses sont combinées, le symbole "+−" est employé.

Choix des comportements de base

Finalement, quand on essaye d'obtenir une nouvelle politique par une combinaison, il faut définir l'ensemble des comportements de base à utiliser. L'ensemble employé ici évolue au cours des essais.

Dans le cas présent, en effet, les tests ont été conduits dans l'ordre présenté par la table et, en commençant avec un ensemble vide de *bb*s (comportements de base), de nouveaux comportements considérés comme "de base" (en lettres **grasses** dans la table) ont été progressivement ajoutés. Il s'agit en l'occurence des deux comportements de base intuitifs déjà utilisés : `éviter` les trous $(0 + 1, -)$, et `pousser` les tuiles dans les trous $(1 + 1, +)$.

7.3.2 Résultats et Analyse

Deux types d'informations peuvent être trouvées dans la table 7.1, lesquels résument nos tests :
- Le *gain moyen* pour 10 000 pas de simulation est l'information principale. C'est la première donnée dans chaque colonne.
- Le *nombre approximatif de séries* (de 10 000 pas de simulation) effectuée avant d'atteindre la meilleure politique est une seconde information utile fournie. Ce nombre est entre parenthèses quand il est disponible (il n'est pas disponible quand on reste à 0).

Remarque : avant de considérer cette table, n'oublions pas que les premiers zéros concernent un agent qui ne cherche qu'à éviter des trous, et qui ainsi ne reçoit aucune récompense positive quand il pousse une tuile dans un trou.

TAB. 7.1 – Table comparative entre des politiques obtenues à partir de rien, par combinaison, et par l'approche incrémentale.

Légende :
- comb. : combinaison de politiques
- inc. : approche incrémentale

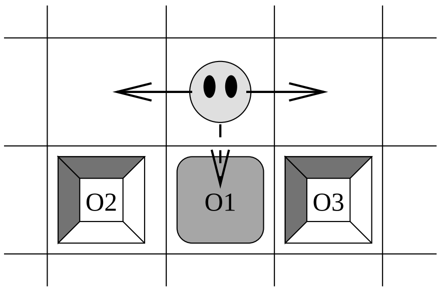
situation		récompense (pour 10 000 pas)				
#tuiles+trous	$\mathcal{R}^c$	table rase		comb.	inc.	
0 + 1	−	0		-800	**0**	
0 + 2	−	0		0	0	
1 + 1	+	1380	(20)	70	**1380**	(20)
1 + 1	+−	150	(300)	1250	1380	(20)
1 + 2	+	1650	(20)	950	1660	(20)
1 + 2	+−	0		380	[0]	
2 + 1	+	1230	(250)	30	1280	(40)
2 + 1	+−	0		30	1250	(40)
2 + 2	+	1250	(200)	250	1260	(80)
2 + 2	+−	0		150	[0]	

7.3.2.1 Chute dans des optima locaux

Si l'on ne prend en considération que les données de la colonne “à partir de zéro”, le phénomène de l'apprentissage classique tombant dans des optima locaux quand les deux récompenses sont en compétition (“+−”) apparaît distinctement, l'agent ne réussissant qu'à éviter les trous (zéros entre crochets). Sur n'importe laquelle de ces lignes “+−”, la combinaison de comportements fournit toujours de meilleurs résultats et permet de pousser une tuile dans un trou plus ou moins souvent. Cela n'amène pas de récompenses moyennes impressionnantes, mais produit comme escompté une bonne politique de départ pour l'apprentissage incrémental.

Des cas de rechute

Seules les situations considérant deux trous et au moins une tuile amènent à un échec. Ceci est probablement dû à une paramétrisation de l'apprentissage qui ne laisse pas assez de place à l'exploration. Un autre fait suggère cette hypothèse : quand l'agent ne cherche qu'à apprendre à éviter un trou, il n'y passe généralement que moins de dix fois pendant tout son apprentissage (il est vite dissuadé de recommencer). La figure 7.4 rappelle à quelle situation correspond le premier cas de rechute.



Quand 2 trous doivent être gérées, comme ici, la politique à adopter va amener l'agent soit à passer dans les trous dans certaines situations, soit à suivre des plans complexes pour débloquer la tuile (d'autant que des cases interdites peuvent la bloquer).
Avoir un apprentissage avec une exploration suffisante pour résoudre ce cas conduirait à une convergence très lente.

FIG. 7.4 – 1^{er} cas de rechute.

On retiendra néanmoins que, outre ces problèmes de rechute, les combinaisons obtenues permettent dans un certain nombre de cas de dépasser de piètres optima locaux.

7.3.2.2 Temps de calcul

Des algorithmes tous coûteux

Un point gênant à propos de la combinaison de comportements est que son optimisation (ici par un recuit simulé) requiert de bonnes estimations de la récompense moyenne gagnée avec un ensemble donné de paramètres. Néanmoins, dans notre situation, le coût de cette phase d'optimisation est négligeable en comparaison avec l'apprentissage complet d'une politique par une montée de gradient. C'est d'autant plus important que la taille de l'espace d'observation dans ce dernier algorithme est ici approximativement multiplié par 16 à chaque nouvel objet considéré, comme le montre la table 7.2.

TAB. 7.2 – Tailles des différents espaces d'observation

objets #tuiles+trous	taille de espace d'observation	nombre de configurations
1 + 1	$16^2/4 = 64$	$1+1=2$
2 + 1	$16^3/4 = 1024$	$1+2=3$
1 + 2	$16^3/4 = 1024$	$2+2=4$
2 + 2	$16^4/4 = 16384$	$2+4=6$

La table ci-contre donne dans différentes situations les tailles approximatives des espaces d'observation rencontrés, la division par quatre étant rendue possible par invariance du problème par rotation.

La dernière colonne indique le nombre de configurations impliquées dans chaque cas, nombre calculé comme étant la somme du nombre d'instances de `[évite trou]` et du nombre d'instances de `[pousse tuile dans trou]`, à supposer que soient utilisés les deux comportements de base intuitifs habituels.

Quelques données

Dans toutes les expériences, on a fixé les durées des différentes phases (un critère d'arrêt selon la stabilisation de l'apprentissage étant difficile à mettre en œuvre) :

- La phase d'adaptation de la combinaison dure $400 * 10\,000$ pas de simulation (même si l'optimum était atteint tôt), c'est-à-dire en moins d'une minute sur l'ordinateur utilisé.
- Le nombre de pas de simulation pour apprendre des politiques complètes (en partant de zéro aussi bien qu'incrémentalement) est aussi fixé, mais à $300*10\,000$ seulement (ces deux durées n'ont pas été réglées de manière précise). Une telle phase dure moins d'une minute pour 1 objet, à peu près 5 minutes pour 2 objets et environ 1 heure pour 3 objets.

Gain en vitesse

Si on ne prête attention qu'aux temps avant d'atteindre le meilleur niveau (notés entre parenthèses dans la table 7.2), on voit que dans la plupart des cas le gain est très important, puisque dans la colonne "table rase" on a une moyenne vers 200, alors que dans la colonne "inc." elle se situe vers 40. L'agent passe donc beaucoup moins de temps dans l'algorithme d'apprentissage par renforcement s'il suit la méthode incrémentale.

Evidemment, cette méthode incrémentale nécessite aussi une phase préliminaire d'optimisation d'une combinaison, laquelle a aussi son coût. Dans le cas du monde des tuiles qui illustre notre discussion, on a vu que c'était plus l'algorithme OLPOMDP qui prenait du temps que la simulation elle-même (et le recuit simulé). Il est donc ici clairement avantageux d'employer la méthode incrémentale que nous proposons.

Vu le nombre d'expériences requis pour obtenir un comportement utilisable, nous sommes de toute façon encore loin de pouvoir nous passer d'un simulateur.

Dernière remarque

La comparaison que nous avons faite entre un apprentissage "table rase" et l'apprentissage incrémental correspondant (sur la même ligne) doit être prise avec quelques précautions. En effet, il ne faut pas oublier que d'autres apprentissages (ceux des comportements de base) ont dû être conduits auparavant. Et encore ne doit-on pas oublier les apprentissages qui n'auront mené à rien d'intéressant...

7.4 Conclusion

Nous sommes partis dans ce chapitre de l'idée que la combinaison de comportements de base fournit une politique qui, si elle n'est pas toujours très efficace, est en général très proche d'une bonne politique (seules quelques décisions ayant réellement besoin d'être changées).

Les expériences effectuées confirment que, dans un certain nombre de cas, il est intéressant de se servir d'un comportement scalable comme d'un exemple pour commencer un apprentissage par renforcement complet. Toutefois, le bilan reste mitigé puisque les cas restant voient l'agent retomber dans une politique se limitant à éviter des trous.

Un autre argument discutable est celui du temps de calcul : l'approche incrémentale est certes plus rapide pour *un* apprentissage donné, mais suppose l'apprentissage de comportements plus simples auparavant. Toutefois, si l'on cherche à concevoir un agent "scalable", comme c'est notre cas, il faut passer par toutes ces étapes, afin de déterminer au fur et à mesure ceux qui peuvent servir de comportements de base.

Revenons à nos moutons

Pour en revenir justement à la conception d'un agent scalable, un problème resté en suspend dans le chapitre précédent est de déterminer quels comportements simples peuvent jouer le rôle de comportements de base (et d'apprendre les dits comportements). Or on a maintenant une méthode qui permet d'apprendre un comportement simple de manière assez efficace sur la base de ceux qui sont déjà connus. On va donc pouvoir s'intéresser dans la suite de nos travaux au moyen d'obtenir cet ensemble de combinaisons de base.

Chapitre 8

Développement d'un arbre de comportements de base

Sommaire

En introduisant au chapitre 6 une méthode de combinaison de comportements, le but suivi était d'obtenir un agent "scalable", c'est-à-dire qui puisse agir de manière efficace dans des environnements de différentes tailles, sachant qu'il sera généralement confronté à de multiples problèmes simples simultanés.

Les premiers essais ont montré que certaines situations s'avéraient bloquantes, une simple combinaison de comportements n'apportant pas nécessairement de bonne solution. Mais si seuls quelques cas particuliers posent problème, peut-être faut-il en faire eux-aussi des comportements de base à utiliser dans les combinaisons ? C'est en suivant ce principe que le chapitre qui commence va pouvoir s'attaquer enfin à l'automatisation de la sélection de comportements de base.

Cette idée nous ramène en fait à un point important de la conception d'agents : l'autonomie (au sens de Russell & Norvig, comme présenté en section 1.1.2). Si l'on a jusqu'à présent fourni à l'agent des comportements de base choisis de manière intuitive (un par objectif), ce n'est apparemment pas un bon choix, et il serait de toute manière préférable que le concepteur n'ait pas à intervenir ici (autrement qu'en définissant les fonctions de récompense).

Ce chapitre s'organise assez simplement en une première section 8.1 présentant la méthode que nous proposons d'adopter pour concevoir l'ensemble de comportements de base cherché, et en une section 8.2 l'expérimentant (avant de conclure en section 8.3).

8.1 Développer un arbre de Comportements de Base

La question soulevée ici est de savoir comment rendre automatique le choix de comportements de base pour la combinaison. Comme cela vient d'être rappelé, c'est une question d'intérêt majeur puisque la sélection intuitive de comportements de base "minimalistes" n'est pas suffisante et parce que, de plus, cette étape est requise pour que l'agent soit réellement autonome.

8.1.1 Principe

Notre proposition est de procéder à la construction d'un ensemble de comportements de base de manière *incrémentale*. Dans ce but, en commençant avec un ensemble $\mathcal{B}$ de comportements de faible complexité déjà sélectionnés, on ajoute progressivement des comportements plus complexes, en utilisant les comportements de $\mathcal{B}$ pour concevoir les nouveaux. Le problème qui vient fort logiquement est de parcourir les comportements dans un ordre qui corresponde à une complexité croissante et à définir un critère pour ne retenir que ceux jugés utiles.

8.1.2 Défrichons le terrain

Avant d'en arriver à un quelconque algorithme, commençons par défricher le terrain en nous intéressant d'abord à cette notion de complexité des comportements.

Complexité des comportements

La définition d'un comportement simple (voir section 6.2.1), comme les travaux conduits jusqu'ici, montrent implicitement que la "complexité" d'un comportement est étroitement liée à deux caractéristiques :

– Le *type de configuration* : Ajouter un objet complique la gestion d'une configuration, puisque les observations et évolutions possibles se multiplient. Depuis un type de configuration donné, les modifications qui vont être possibles correspondent à l'ajout d'un type d'objet parmi les n_{to} types possibles.

Cette croissance peut d'ailleurs se faire indéfiniment, ce qui va nous pousser à considérer cette dimension "type de configuration" comme prioritaire.

– La *combinaison de récompenses* : Souvent, on parlera plus simplement de "récompense". On a vu au cours des chapitres précédents l'intérêt qu'il y a à distinguer les récompenses élémentaires. Cela permet éventuellement de commencer par résoudre des problèmes simples en considérant les récompenses séparément.
  Ici, à l'opposé du cas des types de configuration, pour un nombre de sources de renforcement n_r limité, on a au plus $2^{n_r} - 1$ combinaisons de renforcement (en excluant la combinaison d'aucune fonction de récompense).

On va pouvoir définir deux relations d'ordre (partielles) correspondant à ces deux dimensions :

Définition 11 (Relation d'ordre partielle sur les types de configuration) *Soient deux types de configuration* $\mathcal{C}_a^T = \langle O_1^T, O_2^T, \dots O_{n_a}^T \rangle$ *et* $\mathcal{C}_b^T = \langle O_1'^T, O_2'^T, \dots O_{n_b}'^T \rangle$. *On dira que* $\mathcal{C}_a^T$ *est plus petit que* $\mathcal{C}_b^T$ *(et on notera* $\mathcal{C}_a^T \subseteq \mathcal{C}_b^T$*) si et seulement si tout type d'objet rencontré dans* $\mathcal{C}_a^T$ *l'est aussi dans* $\mathcal{C}_b^T$*, et ce, en aussi grand nombre au moins (s'il y a deux trous dans l'un, il en faut au moins deux dans l'autre).*

Définition 12 (Relation d'ordre partielle sur les combinaisons de récompenses) *Soient deux combinaisons de récompenses* $\mathcal{C}_a^R = \langle r_1^T, r_2^T, \dots r_{n_a}^T \rangle$ *et* $\mathcal{C}_b^T = \langle r_1'^T, r_2'^T, \dots r_{n_b}'^T \rangle$. *On dira que* $\mathcal{C}_a^R$ *est plus petit que* $\mathcal{C}_b^R$ *(et on notera* $\mathcal{C}_a^R \preceq \mathcal{C}_b^R$*) si et seulement si toute récompense élémentaire de* $\mathcal{C}_a^R$ *est aussi dans* $\mathcal{C}_b^R$.

De là, on définit assez simplement une troisième relation d'ordre partielle, sur les paires $(\mathcal{C}^T, \mathcal{C}^R)$ (que l'on assimilera parfois avec le terme "comportement") :

Définition 13 (Relation d'ordre partielle sur les paires $(\mathcal{C}^T, \mathcal{C}^R)$) *Soient deux paires* $\mathcal{C}_a^{T,R} = (\mathcal{C}_a^T, \mathcal{C}_a^R)$ *et* $\mathcal{C}_b^{T,R} = (\mathcal{C}_b^T, \mathcal{C}_b^R)$. *On dira que* $\mathcal{C}_a^{T,R}$ *est plus simple (moins complexe) que* $\mathcal{C}_b^{T,R}$ *(et on notera* $\mathcal{C}_a^{T,R} \sqsubseteq \mathcal{C}_b^{T,R}$*) si et seulement si* $\mathcal{C}_a^T \subseteq \mathcal{C}_b^T$ **et** $\mathcal{C}_a^R \preceq \mathcal{C}_b^R$ *sont vérifiés.*

Où est l'arbre ?

Si l'on tient compte principalement du type de configuration, ceci amène à la réalisation d'un *arbre* de comportements de complexité croissante, chaque nœud correspondant à un type de configuration, ses fils étant augmentés d'un objet parmi les types possibles. La figure 8.1 montre cet arbre d'exploration de types de configurations.

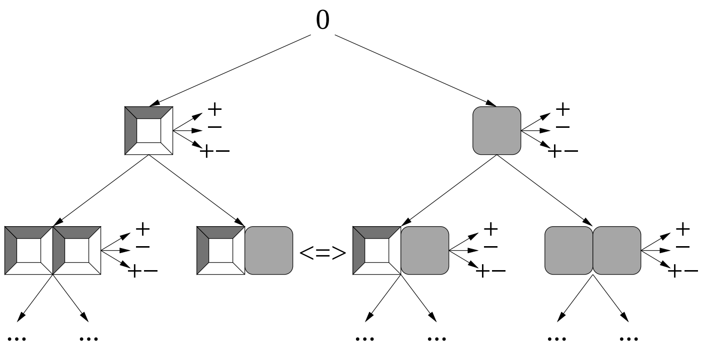
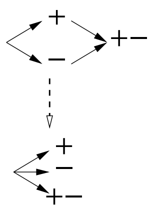


FIG. 8.1 – Arbre d'exploration des comportements.

On a représenté sur cette même figure les différentes combinaisons de récompenses qui vont être prises en compte (toujours symbolisées par les signes "+", "−" et "+−"). Voir à ce propos la figure 8.2.



Comme le montre la partie haute de la figure ci-contre, les combinaisons de récompense s'organisent elles aussi en arbres (en fait en graphes orientés acycliques, puisqu'un nœud peut être rejoint par divers chemins).
Pour des raisons de commodité, c'est le schéma du bas qui est utilisé dans ce manuscrit.

FIG. 8.2 – Arbre d'exploration des comportements.

Pour résumer le principe suivi, un nouveau type de configuration étant choisi, les comportements correspondant aux récompenses possibles sont appris et testés de manière à déterminer s'ils sont intéressants ou pas. La figure 8.1 illustre la progression selon les types de configuration et les récompenses : les comportements sont de plus en plus complexes quand on descend vers les feuilles de l'arbre (même si ce n'est pas un arbre au sens strict, mais plutôt un graphe orienté acyclique à deux dimensions : types de configuration et combinaisons de récompenses).

8.1.3 Croissance arborescente

Parcours

Pour que la construction de l'ensemble de comportements de base $\mathcal{B}$ se fasse en suivant une complexité croissante, l'arbre à explorer doit être parcouru en *largeur d'abord* : on examine toutes les situations impliquant au plus n_{to} types d'objets avant de passer à $n_{to}+1$.

Dans le même ordre d'idée, pour chaque type de configuration, on effectue un parcours en largeur d'abord dans l'arbre des combinaisons de récompenses (tel que représenté dans la partie haute de la figure 8.2).

Exploration (problème)

A supposer qu'on ait défini sur quel critère un comportement va être retenu comme comportement de base, il n'a pas encore été dit précisément quels comportements vont être testés.

Ayant principalement travaillé sur l'application du monde des tuiles, il est facile, pour un type de configuration donné, d'explorer systématiquement toutes les combinaisons de récompenses. C'est donc le choix qui a été fait, ce qui ramène le problème de l'exploration à la dimension principale des types de configuration.

N'ayant plus à tenir compte que d'une dimension (c'est-à-dire un arbre simple), on se retrouve en fait avec un problème similaire à celui du développement d'un *U-tree* auquel s'est intéressé [McCallum, 1995] (ou d'un observable si l'on se réfère à [Dutech et Samuelides, 2003]). Il s'agit en effet de partir d'un arbre de nœuds explorés, et de l'étendre.

Exploration (solution)

Pour cela, quand un comportement lié à un nœud n est retenu pour faire partie de $\mathcal{B}$, on développe localement un sous-arbre *à explorer*. Ce choix se justifie par l'idée que s'il s'avère intéressant d'apprendre un comportement donné, c'est qu'il devait poser problème, et que des situations plus complexes (donc qui en dérivent) pourraient aussi être utilement considérées.

En pratique, cela requiert la définition d'un premier paramètre : la profondeur p_{sa} d'un tel sous-arbre. L'algorithme va ainsi créer continuellement une “frange” de nouveaux types de configuration à évaluer, au fur et à mesure de l'exploration elle-même. Ce processus s'arrête dans deux cas :
- soit l'arbre a été complètement exploré,
- soit une profondeur limite $p_{\max}$ a été atteinte (pour ne pas suivre d'éventuelles branches infinies).

Critère

Sachant comment va être effectuée l'exploration de comportements, il ne reste qu'à définir selon quel critère un comportement sera retenu pour faire partie des compotements de base servant à l'agent. Pour cela, un critère simple d'ajout d'un comportement à “l'ensemble de base” peut être établi sur la comparaison entre les efficacités de la politique scalable (V_{sca}) et du nouveau comportement (V_{nb}) appris.

L'hypothèse que nous suivons est qu'un comportement significativement meilleur que la politique scalable correspondante (celle qui a servi de base à son apprentissage) est susceptible :
- d'une part d'améliorer l'efficacité de l'agent dans la situation à laquelle il répond (ce qui est logique), et
- d'autre part d'amener des informations utiles à d'autres comportements qui reposeraient sur lui (c'est-à-dire dont les paires $(\mathcal{C}^T, \mathcal{C}^R)$ sont plus complexes).

Pour donner un exemple de critère, dans nos expérimentations[49] (voir section 8.2), un comportement est choisi comme nouveau comportement de base si :
- $|V_{sca} - V_{nb}| > 500$
  (ce qui évite des erreurs dues à deux estimations proches l'une de l'autre) et,
- soit $((V_{sca} < 0)$ et $(V_{nb} > 0.9 * V_{sca}))$
  (une faible amélioration d'un résultat négatif suffit)
- soit $((V_{sca} \geq 0)$ et $(V_{nb} > 2 * V_{sca}))$
  (une amélioration majeure d'un résultat positif est requise).

On peut ré-écrire ce critère sous forme algorithmique comme suit dans l'algorithme 6, que l'on notera par la suite $Critere(V_{sca}, V_{nb})$.

La définition de ce critère est sujette à discussion. Les paramètres des formules détaillées ci-dessus, par exemple, ont ici été définis manuellement, et peuvent dépendre de l'application. Mais nous reviendrons sur ce sujet dans l'analyse de nos expériences.

8.1.4 Algorithme

Avant de la tester, nous présentons d'abord une définition formelle de notre méthode dans l'algorithme 7. Ses premières lignes rappellent les données qu'il requiert.

Le processus de construction de l'ensemble $\mathcal{B}$ de comportements de base étant maintenant assez bien détaillé, on va pouvoir passer dans la section suivante à son évaluation.

[49](où l'efficacité est le gain total en 100 000 pas de simulation)

Algorithme 6 Critère suivi pour retenir un comportement dans $\mathcal{B}$

Entrées: V_{sca} : efficacité du comportement scalable.
V_{nb} : efficacité du nouveau comportement simple appris.

```
Critere = faux
si |V_sca − V_nb| > 500 alors
    si ((V_sca < 0) et (V_nb > 0.9 * V_sca))
    ou ((V_sca ≥ 0) et (V_nb > 2 * V_sca)) alors
        Critere = vrai
    fin si
fin si
```

Sorties: $Critere$

Algorithme 7 Un arbre croissant de comportements de base

Entrées: $Critere(\cdot,\cdot)$: pour évaluer les nouveaux comportements.
$p_{\max}$: profondeur maximale de l'arbre, et p_{sa} : profondeur des arbres-fils développés.

```
T arbre vide.
Ajouter à T tout comportement avec de 0 à p_sa types d'objets.
pour tout  Comportement b encore à visiter (jusqu'à la profondeur p_max) faire
    Adapter la combinaison des comportements de base actuels (les b_[basic]) :
    V_sca ← efficacité de cette combinaison.
    Apprendre b par une recherche directe de politique (initialisée par la combinaison) :
    V_nb ← efficacité de ce nouveau comportement.
    si  Critere(V_sca, V_nb) = vrai alors
        Ajouter à T les fils de b ayant jusqu'à p_sa types d'objets en plus.
        Marquer b comme [basic].
    fin si
    Marquer b comme [visité].
fin pour
```

Sorties: Un ensemble $\mathcal{B}$ de comportements de base retenus.

8.2 Expériences

Comme à notre habitude, nous commençons par décrire en section 8.2.1 la méthode employée lors de nos tests, avant de présenter et analyser les résultats obtenus dans la section 8.2.2 qui suit.

8.2.1 Méthodologie

Les expérimentations conduites ont simplement consisté en deux parties :

1. Appliquer l'algorithme de croissance arborescente pour produire un ensemble de comportements de base $\mathcal{B}_{arbre}$. Les paramètres utilisés sont ici :
   – un monde 6×6, avec
   – une profondeur d'exploration maximale $p_{\max} = 4$,
   – une profondeur de développement $p_{sa} = 1$ et
   – le critère détaillé en section 8.1.3 page 177.
2. Tester son efficacité dans diverses situations (dans un monde 8×8).

Dans la seconde partie des expérimentations, on utilise $\mathcal{B}_{arbre}$ avec différents nombres de tuiles et de trous dans l'environnement (jusqu'à cinq de chaque). Cette combinaison est comparée avec une combinaison employant un arbre de référence $\mathcal{B}_{ref}$ de comportements de base "intuitifs" (les deux utilisés dès le chapitre 6 : **éviter** un trou et **pousser** une tuile dans un trou). L'efficacité d'une combinaison est mesurée à travers le gain total reçu durant 100 000 pas de simulation.

8.2.2 Résultats et analyse

8.2.2.1 Croissance arborescente

Examinons d'abord le résultat immédiat de l'algorithme de croissance arborescente. La figure 8.3 montre l'arbre développé des comportements qui ont été évalués. Les signes mis entre parenthèses indiquent que le comportement simple lié à cette combinaison de récompenses et au type de configuration correspondant a été retenu comme comportement de base.

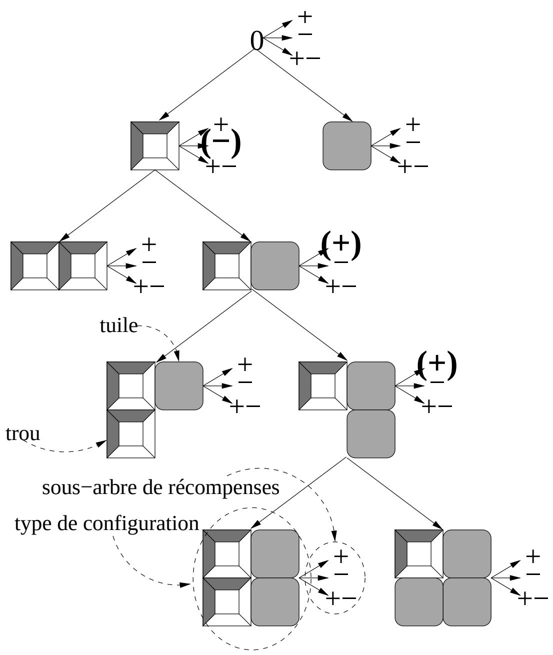
A propos de 2-tuiles/1-trou

Les trois comportements de base obtenus (ensemble $\mathcal{B}_{arbre}$) correspondent aux deux comportements intuitifs (de $\mathcal{B}_{ref}$) et au comportement "spécialisé" dans la résolution du blocage présenté en figure 6.6.

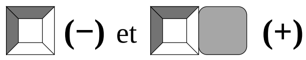
Ce nouveau comportement de base ne tient compte que de la récompense élémentaire positive $(2 + 1, +)$. Il permet toutefois d'améliorer le score obtenu *par combinaison* pour le même type de configuration et les deux récompenses élémentaires $(2 + 1, +-)$. L'apprentissage incrémental ne permet d'ailleurs pas d'améliorer le score obtenu dans cette situation.

A propos de 1-tuile/2-trous

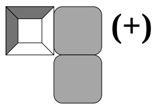
L'apprentissage incrémental réussit aussi à résoudre le problème de la figure 6.7, c'est-à-dire le comportement du nœud 1-tuile/2-trous avec la récompense "+" $(1 + 2, +)$. Toutefois, l'augmentation de performance qui s'ensuit ne répond pas au critère ($V_{nb} \simeq 1.5 * V_{sca} < 2 * V_{sca}$) et, en conséquence, ce comportement simple n'est pas ajouté à $\mathcal{B}_{arbre}$. Ce point mis à part, l'arbre généré est plutôt satisfaisant.



En exécutant l'algorithme proposé, aux deux comportements que l'on peut intuitivement juger comme suffisants pour le monde des tuiles :



vient s'ajouter un comportement résolvant l'un des blocages qui était apparu dans nos essais au chapitre 6 :



.

FIG. 8.3 – L'arbre des comportements testés et (entre parenthèses) ceux qui ont été retenus.

Note : Comme dans la table 7.1, le cas $(1 + 2, +-)$ voit un phénomène de rechute, le comportement scalable obtenu n'aidant pas à atteindre un bon comportement.

Autres essais

Une des conditions du critère présenté en section 8.1.3 peut paraître assez stricte, puisqu'il s'agit de doubler au moins le score atteint par combinaison des comportements de base déjà connus ($V_{nb} > 2 * V_{sca}$). On a donc lancé le même algorithme avec une version du critère qui ne requiert qu'une faible amélioration des résultats positifs ($V_{nb} > \mathbf{1.3} * V_{sca}$).

Malheureusement, l'algorithme trouve alors de bien trop nombreux comportements éligibles (huit ont été enregistrés dans un cas où $p_{\max} = 4$). En observant les comportements des agents, on se rend en fait compte qu'une combinaison de comportements ne perd pas de temps *que* dans des situations bloquantes telles que celles qu'on a déjà vues, mais aussi dans des situations *indécises*. En effet, si l'agent hésite entre deux objectifs de poids équivalents (deux tuiles possibles à pousser dans le même trou), il risque d'osciller.

L'apprentissage incrémental permet de lever ces hésitations, l'agent optant pour l'un des deux objectifs aux dépens de l'autre. Mais on peut voir là un biais de programmation : malgré l'utilisation de symétries du problème, le fait qu'on traite les perceptions systématiquement de la même façon (la source de nourriture N_1 avant N_2 si l'on se réfère à la figure 8.4) amène une distinction entre objets de même type. Ce défaut crée implicitement un effet de mémoire : si le but de l'agent est d'aller chercher N_2 à l'instant t, il restera le même à $t + 1$. On a de manière indirecte introduit un phénomène de *persistance de l'attention* qui permet à l'agent de garder un but à plus long terme.

Pour conclure sur ce point, on a choisi de se concentrer sur le critère présenté en section 8.1.3, puisqu'il a l'avantage de sélectionner un nombre limité de comportements de base, et ce, sans dégradation majeure quand on mesure l'efficacité de la combinaison.

a) La combinaison amène à hésiter.

Quand on combine deux comportements (figure a), l'agent hésite et se met à osciller.

b) Résultat après apprentissage.

Quand on apprend un unique comportement simple (figure b), si les perceptions permettent de reconnaître une source de nourriture entre t et $t+1$, alors une persistance de l'attention apparaît, laquelle permet à l'agent de persévérer dans ses décisions.

FIG. 8.4 – Apparition d'un phénomène de persistance de l'attention.

8.2.2.2 Efficacité de l'arbre

Après la génération d'un arbre de comportements et surtout la sélection d'un ensemble de comportements de base $\mathcal{B}_{arbre}$, on va pouvoir évaluer son efficacité en utilisation.

Critère "normal"

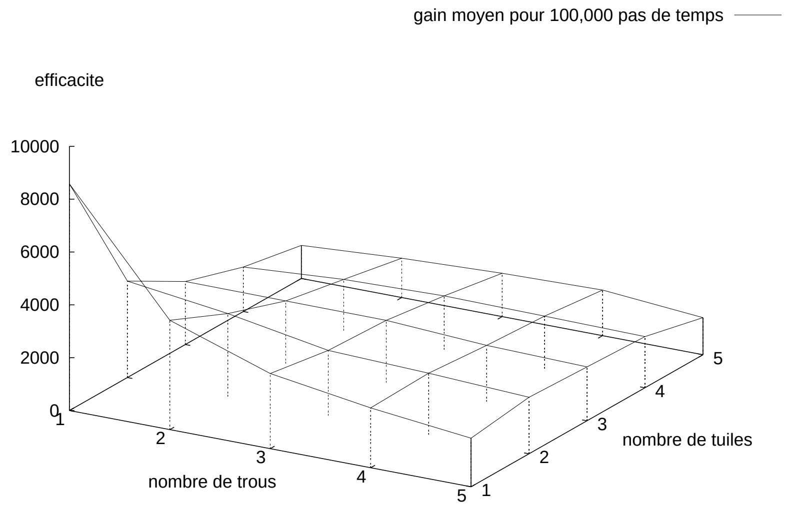
Les figures 8.5 a) et b) présentent l'efficacité de deux ensembles de comportements de base : $\mathcal{B}_{ref}$ et $\mathcal{B}_{arbre}$, de manière à comparer ce dernier (automatiquement généré par notre algorithme) avec le premier (conçu à la main). Les axes x et y du plan horizontal indiquent le nombre de trous (abscisses) et de tuiles (ordonnées) dans les situations évaluées (les nombreux objets à manipuler ont obligé à étendre la grille utilisée à une taille de 8×8). Sur l'axe z est mesuré l'efficacité du comportement obtenu dans chaque situation.

A première vue, les deux surfaces sont assez similaires, avec un pic commun dans la situation simple 1-tuile/1-trou (à laquelle le comportement de base "pousser" est dédié). Comme $\mathcal{B}_{ref}$ et $\mathcal{B}_{tree}$ ne diffèrent que par le comportement gérant la récompense "pousser" pour 2-tuiles/1-trou, les bons résultats avec 1 trou et de nombreuses tuiles sont attendus. Néanmoins, cette augmentation est importante et semble persistente avec un nombre croissant de tuiles à pousser (les résultats sont multipliés par 2) le long de l'axe des tuiles.

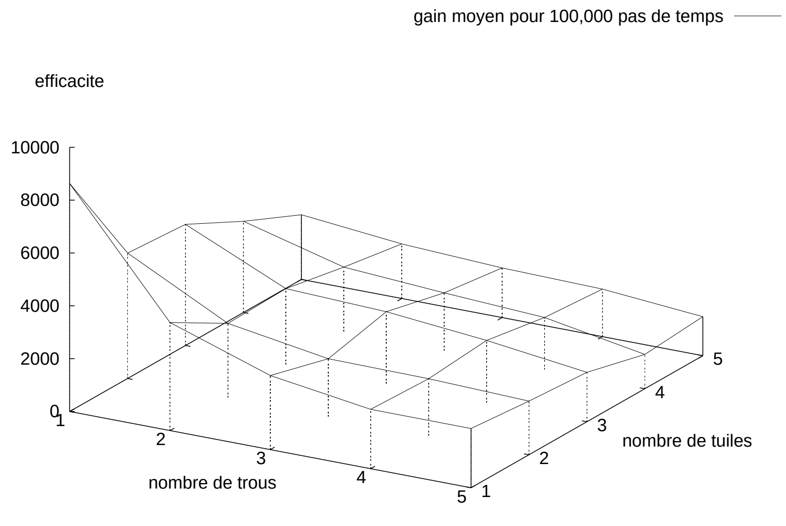
Si l'on considère un nombre croissant de trous, les résultats de la figure b) sont de plus en plus proches de la surface de référence (figure a)). Les variations subsistantes sont partiellement dues à des erreurs d'estimation de l'efficacité. Nous avons en outre vu qu'un problème lié au critère empêche d'apporter des améliorations dans ces cas à deux trous et plus.

Critère assoupli

La figure 8.5 c) montre les résultats obtenus avec un critère "moins strict" ($V_{nb} > \mathbf{1.3} * V_{sca}$) et une profondeur d'arbre maximale $p_{\max} = 3$. On note que le comportement de base ainsi ajouté $(1+2, +)$ permet de nettement améliorer les résultats pour une tuile et plusieurs trous, et d'avoir un léger gain dans les cas à plus d'une tuile.



a) comportements de base issus de $\mathcal{B}_{ref}$



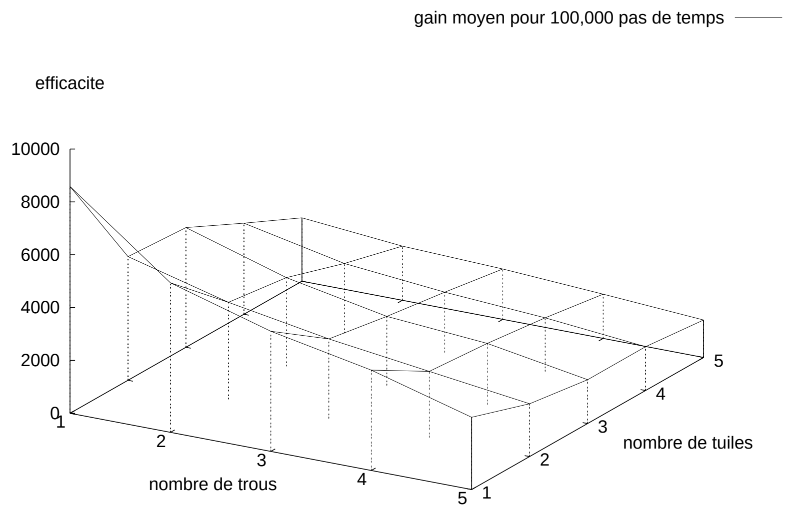
b) comportements de base issus de $\mathcal{B}_{arbre}$

Les ensembles de comportements de base sont :

a) $\mathcal{B}_{ref}$: les deux comportements intuitifs `pousse` et `évite`,

b) $\mathcal{B}_{arbre}$: les trois comportements obtenus avec le critère "normal" (voir figure 8.3), et

c) $\mathcal{B}'_{arbre}$: les trois mêmes comportements complétés du comportement de base considérant 2 trous et 1 tuile pour la simple récompense "+".



c) comportements de base issus de $\mathcal{B}'_{arbre}$

FIG. 8.5 – Efficacité de la combinaison dans des situations plus complexes avec différents ensembles de comportements de base.

8.2.3 Bilan

Pour résumer l'analyse de ces expérimentations, il y a apparemment un compromis à trouver entre un nombre réduit de comportements de base et de hautes performances. De plus, alors que cette approche cherche à automatiser le choix de comportements de base, l'algorithme de croissance arborescente implique divers réglages parmi lesquels $p_{\max}$, p_{sa} et le critère. Une suggestion pour améliorer la sélection de comportements de base est de suivre des critères plus contraignants quand la complexité des comportements croît.

Rappelons que les outils d'apprentissage par renforcement classiques ne seraient pas capables de gérer plus d'une tuile et un trou simultanément. Ils seraient réduits à trouver comment éviter de tomber dans les trous. Mesurer leur efficacité comme on l'a fait en figure 8.5 produirait un pic similaire pour 1-tuile/1-trou, le reste de la surface restant à un niveau nul.

8.3 Conclusion

8.3.1 Contribution de ce chapitre

Après avoir étudié au chapitre 6 les possibilités offertes par une méthode de combinaison de comportements, le chapitre qui se conclut maintenant a proposé un algorithme pour que l'agent soit à même de déterminer de manière autonome les comportements de base dont il doit se servir. Il sait trouver de quelles capacités il a besoin et sait les apprendre (et se sert dans ce but de l'algorithme d'apprentissage incrémental vu au chapitre 7).

Dans le cadre de travail que nous avons adopté, le concepteur a pour tâche principale de définir les fonctions de récompense élémentaires dont va se servir l'agent, ce qui n'est pas une mince affaire, mais reste une tâche incontournable. C'est en évaluant la valeur ajoutée de chaque nouveau comportement simple (comparé aux comportements de base déjà retenus dans $\mathcal{B}$) qu'il est possible de déterminer s'il doit être utilisé pour augmenter l'ensemble $\mathcal{B}$. Ce processus itératif est répété jusqu'à ce qu'il n'y ait plus d'ajout.

8.3.2 Validation

Notre algorithme a été testé sur le problème du monde des tuiles. Il a donné de bons résultats puisqu'il a été capable de proposer un ensemble de comportements qui s'avère plus complet que l'ensemble intuitif conçu à la main. En outre, notons que les problèmes complexes auxquels l'agent était confronté ne pouvaient pas être résolus de manière directe par un apprentissage par renforcement classique.

La capacité de la combinaison de comportements à fonctionner efficacement avec des nombres grandissant d'objets reste limitée (mais les chapitres précédents ont déjà discuté ce point), mais la base de comportements obtenue se montre plus efficace que ceux proposés manuellement. Dans le cadre de l'utilisation d'une méthode par sélection d'action, il semble donc intéressant d'employer de telles méthodes pour déterminer les capacités utiles de l'agent.

Conclusion

Cette troisième partie a permis d'aborder la question de la complémentarité entre systèmes multi-agents et apprentissage par renforcement sous l'angle de l'utilisation d'une approche multi-agents pour concevoir *un* agent en suivant le principe de l'apprentissage par renforcement (en résumé : s'adapter pour maximiser son gain moyen).

Travail accompli

Nous étions aussi attachés à la conception d'agents qui soient “scalables”, c'est-à-dire capables de travailler dans des environnements de tailles variables, le nombre d'objets avec lesquels interagir (et donc les perceptions) pouvant évoluer. Ainsi, nous nous sommes naturellement intéressés à travers le chapitre 5 aux approches par décomposition dans l'apprentissage par renforcement, et à leurs liens avec le domaine de la sélection d'action.

Cette étude nous a permis d'établir l'objectif des chapitres suivants, c'est-à-dire de concevoir une *architecture de sélection d'action* dont les principales caractéristiques sont d'être :

- à flux-libre et produisant une politique stochastique (pour éviter certaines formes de blocages) ; et
- auto-organisatrice, les modules (comme en a toute architecture de sélection d'action) étant développés par apprentissage par renforcement, sélectionnés, et agencés automatiquement.

Le chapitre 6 a permis d'étudier une voie pour concevoir une telle architecture, sans pour autant que l'aspect “auto-organisation” soit complet, puisque les modules étaient encore choisis manuellement. Les résultats mitigés dûs à des blocages dans des cas particuliers ont incité à améliorer par une phase d'A/R les comportements scalables (combinés) obtenus, comme l'a présenté le chapitre 7. Finalement, c'est grâce à cet *apprentissage incrémental* qu'est venue l'idée d'une méthode de développement et de sélection des comportements de base de l'agent, décrite dans le chapitre 8 et fondée sur la croissance d'un *arbre de comportements*.

Si les résultats obtenus dans le monde des tuiles rappellent que les différents algorithmes proposés restent de simples heuristiques, l'objectif fixé est atteint. L'agent nécessite une intervention bien moindre de son concepteur que ce peut être le cas dans les approches de sélection d'action que nous avons rencontrés, puisque le choix et la réalisation des modules ne requiert plus que la définition de fonctions de récompense élémentaires.

En outre, cette méthode qui permet à un agent d'établir de manière autonome une bibliothèque de comportements disponibles est une méthode assez générique, puisque l'approche suivie n'a rien de spécifique au problème du monde des tuiles. On pourrait même s'affranchir de la notion d'objet pour se restreindre à celle de “percept”. Seul le fait que les récompenses doivent être nulles la plupart du temps est réellement particulier, mais on peut souvent se ramener à ce cas.

A propos d'architectures internes

L'architecture obtenue est une architecture interne à première vue assez comparable aux architectures horizontales telles que présentées en section 1.1.5.3, puisque les modules sont indépendants et qu'un contrôleur détermine au final le comportement à adopter.

Néanmoins, dans le fonctionnement de notre architecture, si certains modules restent muets la plupart du temps (c'est le cas des comportements d'évitement), ils peuvent interrompre les autres, ou plutôt les "contraindre" momentanément (si l'agent passe à côté d'un trou). On se rapproche ainsi d'architectures plus complexes avec des modules capables de prendre le pas sur d'autres.

Le lecteur intéressé pourra aussi se référer à [Dury, 2000], puisque le même monde des tuiles y a été abordé à l'aide d'une autre forme d'architecture interne. Il s'agit en effet de définir un comportement à travers des règles d'interaction (une observation déclenchant une action), les dites règles étant organisées entre elles à travers un système de priorités.

Perspectives

Dans l'approche présentée, on crée des agents-comportements (pour reprendre le point de vue multi-agents de ces travaux) ainsi qu'une organisation : on règle les paramètres qui régissent leurs rapports. Un point qui reste problématique est le choix de cette "interaction", c'est-à-dire de la relation entre les agents-comportements. Il n'y a probablement pas de solution idéale pour cela, mais l'étude des relations entre comportements pourrait améliorer les choses (comportement inhibiteur, incitateur, comportements concurrents ou non...).

Cette direction de recherche permettrait d'ailleurs probablement de se rapprocher d'une véritable architecture multi-agents, en rendant l'organisation entre agents-comportements plus souple et adaptative (défaut de notre approche dont nous avions discuté dès la section 5.5).

Une autre perspective très intéressante concerne la gestion des objets perçus. L'idée est de rendre l'agent capable de définir des *types d'objets abstraits*. Actuellement, les types d'objets présents dans l'environnement sont donnés à l'agent. Mais nous pensons que l'on pourrait se servir des données de l'apprentissage pour catégoriser automatiquement les objets en classes (comme par exemple "obstacles", "but", "à bouger", etc). Il s'agirait là d'un pas supplémentaire vers un véritable méta-apprentissage.

Conclusion et perspectives

« L'art de la citation est l'art de ceux qui ne savent pas réfléchir par eux-même. »

Voltaire

Note : Avant de conclure ce mémoire, précisons que les annexes de cette thèse contiennent quelques compléments qui n'ont pas trouvé leur place dans le fil de la thèse. C'est le cas d'une rapide discussion sur le choix de passer par des simulations (section A.1).

1 Bilan

Comme annoncé en introduction, le point de vue de l'intelligence artificielle que nous avons adopté au cours de cette thèse est celui de la conception de systèmes agissant de manière rationnelle (c'est-à-dire pour atteindre un objectif), système que l'on a décidé de désigner par le terme "agent". Cela nous a conduit à considérer en premier lieu le domaine de l'*apprentissage par renforcement*, lequel répond assez bien à cette définition. Mais, étant aussi intéressés par les résultats que l'on peut obtenir à travers des phénomènes émergents, nous avons décidé de travailler aussi dans le champ des *systèmes multi-agents*.

Or chacun de ces deux domaines souffre de certaines difficultés pour lesquelles l'autre domaine peut proposer des solutions. Cette thèse a ainsi été l'occasion d'employer de manière concomitante ces deux domaines de l'intelligence artificielle, dans deux approches symétriques dont nous allons maintenant tirer un bilan.

L'apprentissage par renforcement à l'usage des systèmes multi-agents

Si l'on sait que des agents très simples peuvent, en groupe, effectuer des tâches relativement complexes, on dispose de peu de moyens de spécifier le comportement que ces agents doivent adopter. C'est là qu'employer des méthodes d'apprentissage par renforcement est séduisant, puisqu'il suffit a priori de définir des situations à récompenser pour que les agents s'adaptent tous seuls.

En y regardant de plus près (à travers des travaux antérieurs sur le sujet), nous avons mis en évidence un certain nombre de difficultés : définir des récompenses individuelles qui reflètent l'objectif du groupe, coordonner les actions d'agents ayant des points de vue distincts, modéliser les autres agents pour être capable d'empathie (et ainsi mieux s'adapter à ses congénères). Si certaines avancées ont été faites sur ces différents points, c'est en général en faisant des hypothèses fortes (telles que l'observabilité totale). Il reste donc intéressant de proposer des méthodes pour aider l'apprentissage de ces agents sociaux.

Or un champ de méthodes existe qui correspond justement à cette idée d'apporter une aide : l'*apprentissage progressif*. C'est en fait une chose assez naturelle qui consiste à s'intéresser d'abord à des problèmes simplifiés, pour guider l'apprenant vers la solution à la tâche complexe qu'il doit accomplir. Ayant présenté ce domaine, nous avons proposé de le mettre en œuvre pour la conception de SMA en étant progressif selon deux dimensions :

- la complexité de la tâche (ce qui est important dans un SMA où les évolutions possibles du système sont en très grand nombre), et
- le nombre d'agents et d'objets présents.

Les résultats sur notre problème-jeu de fusion de bloc ont été très encourageants, permettant dans les deux cas d'accélérer l'apprentissage et dans le second cas de dépasser très efficacement quelques optima locaux. En évaluant la première étape de notre approche sur le problème proie-prédateurs[50], on a aussi pu mettre en évidence un défaut de l'apprentissage progressif en cas d'observabilité partielle : il y a un risque de devoir "désapprendre", et donc perdre du temps que l'on croyait gagner.

Les systèmes multi-agents à l'usage de l'apprentissage par renforcement

Une des difficultés que l'on peut rencontrer dans l'utilisation de l'apprentissage par renforcement pour concevoir un agent est la gestion d'objectifs concurrents. Travaillant dans des cadres non-markoviens, on risque en effet de voir l'agent tomber dans des optima locaux consistant à ne tenir compte que des objectifs les plus faciles à atteindre, alors que des comportements un peu plus recherchés pourraient apporter un plus grand gain.

Cette problématique a été mise en rapport avec les travaux sur la sélection d'action. Nous avons ainsi choisi d'avancer vers un processus de sélection d'action qui ait le moins possible besoin de l'intervention du concepteur (grâce à l'apprentissage par renforcement). Dans ce cadre, nous nous sommes intéressés à des environnements de tailles variables, particulièrement en ce qui concerne le nombre d'objets perçus par l'agent.

Ainsi, nous avons d'abord proposé une forme assez générique de combinaison de comportements adaptative, dans l'idée de voir notre agent pondérer les réactions que lui suscitent les différents objectifs potentiels au sein de son environnement. Différentes formes plus spécifiques ont été évaluées, permettant certes d'obtenir des résultats qu'un apprentissage classique n'atteindrait pas (dépassements d'optima locaux comme souhaité) et permettant aussi de réutiliser les poids appris, mais quelques blocages dans des situations particulières dégradent de manière importante les politiques calculées.

Nous avons pu "remédier" à cette dégradation en nous servant du comportement combiné obtenu comme base à un nouvel apprentissage complet de politique. Cette idée a fonctionné dans la limite de la gestion du dilemme exploration/exploitation, puisque dans quelques cas l'agent est retombé dans de mauvais optima locaux. C'est toutefois grâce à cet outil que nous avons élaboré un algorithme générant un ensemble de comportements de base à utiliser pour recombinaison, ce qui constitue un résultat original : nous ne connaissons pas d'autres systèmes de sélection d'action qui fasse de même. Et si la croissance arborescente réalisée requiert quelques réglages, elle a permis de trouver un meilleur choix de comportements que celui effectué intuitivement.

Pour finir

Dans l'une comme dans l'autre partie de notre travail, nous avons abouti à un système dont l'architecture est de type multi-agents, et conçue par apprentissage par renforcement. Si ce n'est le point de vue qui a guidé l'un et l'autre, on se retrouve dans les deux cas avec une société d'agents qui évoluent pour constituer une entité plus complexe. On peut penser ici à la notion de *société de l'esprit* de Minski [Minsky, 1986] (Bien que l'on reste loin d'un système aussi élaboré que le sien). Mais, dans le principe, cette alliance entre SMA et A/R peut aussi être mise en parallèle avec les modèles connexionnistes, dans lesquels un grand nombre d'entités simples apprennent à travailler ensemble.

Mais, comme nous l'avons fait remarquer, l'aspect multi-agents est entaché de défauts aussi bien dans les premiers travaux (puisque la méthode employée implique une forme de centrali-

[50] D'ailleurs un jour il faudra qu'on enseigne à la proie l'art de la fuite.

sation) que dans les seconds (l'organisation des "agents" étant assez rigide). C'est pour cette raison que nous avons préféré parler dans le titre de cette thèse de deux approches *modulaires* de l'apprentissage par renforcement. De plus, un pléonasme nous permet d'insister dans ce titre sur le fait qu'à nos yeux un agent est une entité intelligente, c'est-à-dire adaptative.

2 Perspectives

Les thèmes abordés au cours de cette thèse, et plus précisément les travaux qui ont été conduit pendant sa durée, amènent des perspectives assez diverses, autant par leurs termes (court ou long) que par leurs natures. Sans pouvoir assurer l'exhaustivité de la liste qui suit, voici d'abord quelques sujets de développements qui ont été envisagés dans une continuité directe de ces travaux :

- **Applications** : En changeant éventuellement de forme d'agents (pour passer plutôt à des agents orientés comme décrits en page 197), il serait intéressant de voir comment se comportent sur de vrais robots les politiques fournies par nos approches. A condition d'utiliser les capteurs appropriés, les perceptions que nous utilisons peuvent être obtenues après une phase de prétraîtement. Par contre, il faudra se contenter d'applications sur des problèmes réalistes (la fusion de bloc et le monde des tuiles n'en sont pas, contrairement à proie-prédateurs ou à des tâches de navigation).
- **Modélisation** : Nous ne sommes pas revenu sur ce point, mais apprendre en ligne comme ce fut le cas de tous nos agents (même s'il s'agissait de simulations) ne semble pas être une approche raisonnable. Des travaux visant à mieux exploiter les données acquises par l'expériences existent, et il pourrait même être intéressant d'en profiter pour passer par une modélisation plus élaborée, fondée par exemple sur un historique d'observations et d'actions (si l'on ne tient compte que du problème des observations partielles).
- **Modélisation dans un cadre SMA** : Un point nous ayant incité à éviter l'usage de modèles est le cadre multi-agents qui rend l'environnement (vu par un agent) non-stationnaire. C'est un champ d'exploration très vaste, mais être capable de modéliser les objectifs des autres agents nous semble un objectif à long terme important pour rendre la coordination efficace à travers des capacités d'empathie. Sinon, on risque d'être contraint à retomber dans des approches coûteuses en expérimentations, puisque sans aucune modélisation.
- **Combinaison de comportements plus souple** : Pour en venir à la combinaison de comportements mise en œuvre, celle-ci reste assez rudimentaire. Elle ne profite pas de possibles relations entre les comportements (qu'une tâche se fasse aux dépens d'une autre ou qu'elles puissent au contraire être toutes deux satisfaites), et passe par un contrôleur plutôt dirigiste. Il serait souhaitable de mieux étudier les rapports entre comportements et s'en servir pour que l'organisation entre comportements de base soit plus *auto*-organisée, ce qui nous rapprocherait un tant soit peu d'un système multi-agents.
- **Gestion de ressources** : Une perspective envisageable à court terme est d'utiliser notre combinaison de comportements pour gérer les ressources d'un agent. Supposons un robot qui doit aller recharger ses batteries régulièrement. Si une variable interne représente le niveau de ses réserves énergétiques, celle-ci peut servir à pondérer l'importance du comportement "retourner me charger".

Les développements de cette thèse ont aussi permis de mettre en avant quelques problématiques qui, pour leur part, correspondent à des objectifs à plus long terme. On peut citer principalement les deux suivantes :

- **Extraction de types d'objets abstraits** : En restant dans le domaine de la combinaison de comportements, une idée qui mérite selon nous réflexion est de chercher à extraire des types d'objets plus abstraits que ceux pour l'instant fournis par le concepteur. Il pourrait s'avérer utile de se rendre compte que deux objets ont, dans certaines situations, des rôles équivalents : un trou et un feu peuvent tous deux faire disparaître certains objets, un rocher et un mouton peuvent être rangés dans la même classe "obstacle" dans certaines situations... Cette idée risque hélas aussi de s'avérer très coûteuse en temps de calcul. Il faut donc la considérer avec précaution.
- **Approches progressives** : Un aspect commun aux deux grandes parties de nos travaux est d'avoir apporté des algorithmes d'apprentissage que l'on peut qualifier de progressifs ou incrémentaux. Dans les deux cas, cela a permis de dépasser des optima locaux, mais quelques difficultés ont subsisté. Ce sont à notre avis des approches à approfondir. Il ne peut être que bénéfique de mieux les gérer, qu'il s'agisse du schéma de progression à suivre (l'entraînement) ou du rythme de progression (ne pas brûler les étapes, mais ne pas non plus s'embourber dans des apprentissages en fin de compte inutiles).

En considérant à la fois les aspects "apprentissage progressif" et "utilisation de modèles" que nous venons d'évoquer, on tend vers le point de vue assez récent de l'intelligence artificielle "développementale". Celui-ci se fonde sur l'idée que l'on a jusqu'à maintenant cherché à réaliser rapidement des systèmes intelligents, alors que pour obtenir une entité aussi complexe et efficace que le sont nombre d'animaux, il faut une élaboration bien plus lente et progressive, de nouvelles capacités plus élaborées se basant sur d'anciennes assez rudimentaires. Ce qui est appelé chez [Weng *et al.*, 2002; Weng, 2002] "développement mental autonome" est, d'une certaine façon, l'idée de voir chez des machines un développement comparable à celui d'un humain, par exemple, tel que décrit par Piaget.

Cette perspective assez ambitieuse de l'IA développementale[51] n'en est qu'à ses débuts. Il nous paraît toutefois intéressant de l'avoir en tête, si cela permet de faire avancer des problèmes tels que l'ancrage des symboles [Harnad, 1990] ou l'apparition de la notion de "groupement" [Piaget, 1967]. De tels travaux semblent devoir requérir des architectures assez souples et modulaires, ce qui fait inévitablement penser à nouveau à la société de l'esprit [Minsky, 1986]. Mais en attendant que ces idées se concrétisent, on peut toujours rêver du jour où l'on racontera des contes de fées aux agents artificiels pour les aider à appréhender leur monde (à propos des contes de fées, lire [Bettelheim, 1976]).

[51] On pourrait aussi parler d'IA développe-mental.

Annexes

Annexe A

Considérations diverses

A.1 Simulation

Le fil conducteur de ce mémoire ne nous a pas donné l'occasion de discuter du choix de travailler sur des simulations, alors que celui-ci est tout à fait critiquable. Il peut toutefois se justifier dans le cas présent, en faisant entrer en ligne de compte les arguments suivants :

- Nous n'avions tout d'abord pas de robots disponibles pour travailler en multi-robots (situation qui devrait être réglée sous peu).
- Vu les algorithmes d'apprentissage (sans modèle) employés, l'élaboration d'un comportement en temps réel par essais-erreur serait irréaliste. Même si l'on arrivait à s'affranchir du passage d'un cadre discret à un cadre continu, une telle méthode prendrait beaucoup trop de temps.
- Une idée pour pouvoir travailler sur de vrais robots est d'apprendre un modèle, de manière à ce que le robot puisse élaborer une politique plus efficacement (hors-ligne). Cela doit toutefois rester coûteux en expérimentations. De plus, si on se limite à tenir compte d'observations partielles, on se retrouve dans des cadres où l'environnement, donc le modèle, est perçu comme dynamique. De plus, le fait que plusieurs agents se côtoient vient accroître cette dynamicité.

En outre, tout bien considéré, l'approche "située" à laquelle nous avons fait référence en introduction de ce mémoire est plutôt ambitieuse. Nous avons d'ailleurs naturellement considéré des perceptions simplifiées (faisant apparaître un certain nombre de symboles). Mais ce n'est pas un argument pour passer par une simulation, puisque l'on peut dans bien des cas avoir une phase de pré-traitement des observations qui permet d'obtenir les informations "symboliques" utilisées.

Pour finir, l'utilisation d'un simulateur est assimilable à une technique *avec* modèle. Avoir un modèle complet de PDMPO peut certes permettre l'emploi de méthodes de planification, mais n'est pas possible dans une situation multi-agents (à moins de savoir modéliser les autres agents). Il reste donc préférable dans notre cas d'employer des algorithmes d'optimisation randomisés, c'est-à-dire des simulations.

A.2 Contre-exemple

Pour montrer que les Q-valeurs ne sont pas directement liées à la politique, considérons le PDMPO représenté sur la figure A.1 (On ne suppose pas ici l'utilisation d'un algorithme d'apprentissage par renforcement particulier.). Une unique observation o cache à tout moment l'un des deux états s_1 ou s_2. Les transitions dépendent des deux actions disponibles (a et b), et sont toutes déterministes, excepté quand on choisit l'action b dans l'état s_1 (probabilité p de rester en s_1 et $(1-p)$ d'aller en s_2). La seule récompense est gagnée en cas de transition $(s_1, b) \rightarrow s_1$.

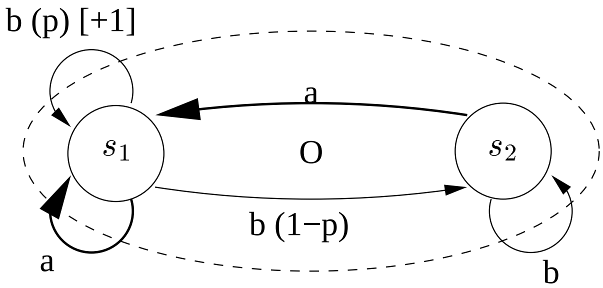


Fig. A.1 – Un exemple de PDMPO dont la politique optimale donne des Q-valeurs assez inattendues.

Supposons que p soit proche de 1. La politique optimale est alors de choisir l'action b la plupart du temps (a est requise de temps en temps, de manière à sortir de l'état s_2), l'état courant étant généralement s_1 :

$$P(o,a) \simeq 0 \quad \& \quad P(o,b) \simeq 1, \tag{A.1}$$

Ainsi $Q^*(o,b) \simeq Q^*(s_1,b)$ est pratiquement la valeur optimale du PDMPO : V^*. Avec un facteur de décompte γ égal à 0.9 (ce qui est une valeur raisonnable), on a :

$$\begin{aligned} Q^*(o,a) &= 0 + \gamma * (P(s=s_1) * V^*(s_1) + P(s=s_2) * V^*(s_2)) && \text{(A.2)} \\ &= \gamma * V^*(s_1) \simeq 0.9 * V^* && \text{(A.3)} \end{aligned}$$

Les deux Q-valeurs ont des valeurs très proches, alors que les probabilités associées aux deux actions sont complètement opposées. Cela montre qu'il n'y a pas de lien fort entre les Q-valeurs et les probabilités d'action.

Si un Q-learning boltzmannien adapté était utilisé sur cet exemple simple, la température pourrait être réglée automatiquement pour atteindre une politique optimale. Mais dans des situations plus complexes, il y a bien plus que cet unique degré de liberté à gérer. On a toutefois gardé la combinaison de Q-valeurs comme un algorithme de référence dans nos expérimentations.

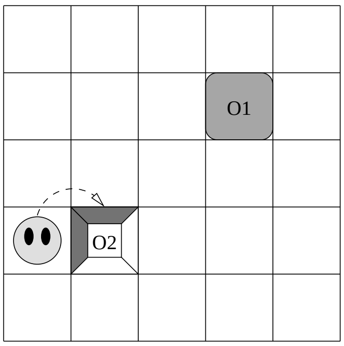
A.3 Quelques problèmes rencontrés avec les problèmes-jeux que nous avons employés

A.3.1 Sur le choix des récompenses

Comme on l'a expliqué dans le chapitre sur l'apprentissage par renforcement, il faut définir soigneusement la fonction de récompense qui va guider l'agent, faute de quoi ce dernier risque de s'avérer astucieux en trouvant un comportement optimal inattendu. Nous avions illustré ce point avec l'exemple de l'agent-aspirateur (voir figure 1.14), lequel redépose de la poussière pour pouvoir en aspirer plus. Et comme nous le racontons ci-après, nous sommes retombés sur cette difficulté en mettant au point le fonctionnement du monde des tuiles qui nous a servi à une bonne partie de nos expériences.

En effet une première version faisait que, quand l'agent passe dans un trou, trou *et* agent sont replacés au hasard quelque part sur le quadrillage de l'environnement (le reste des lois régissant ce petit univers étant comme présenté dans la section 5.3.2). Or, en laissant l'agent apprendre à simplement pousser une tuile dans un trou (sans chercher à éviter le dit trou), on a pu le voir contre toute attente se jeter régulièrement dans le trou de son plein gré.

Ce comportement s'explique assez simplement par le fait que l'agent, quand le trou se trouvait entre lui et la tuile, avait plus de chances de se rapprocher vite de la tuile convoitée par le "télétransport" que constitue le saut dans le trou. Une telle transition est présentée par la figure A.2.



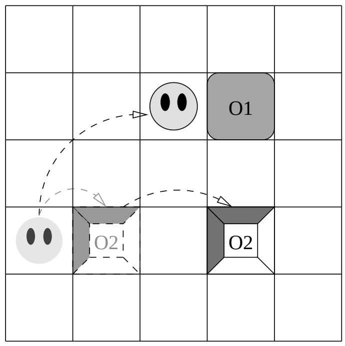


Fig. A.2 – Pourquoi se jeter dans un trou ? Pour se déplacer plus vite...

Nous avons préféré faire disparaître ce phénomène et laisser un agent allant dans un trou en ressortir de son plein gré. Non seulement cela évite d'avoir un comportement qui cherche explicitement à passer dans un trou, mais l'environnement n'en est que plus simple à appréhender.

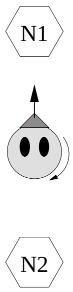
A.3.2 Sur les agents orientés (avec un "devant")

Dans le cadre de nos problèmes-jeux, un autre choix de conception amenant à réfléchir est celui du caractère orienté ou non des agents. Nous avons opté pour des agents *non-orientés* (sans notion de "devant"), les déplacements possibles correspondant aux quatre points cardinaux.

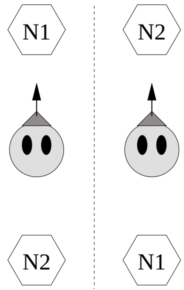
Ce choix permet de simplifier la simulation et d'exploiter la symétrie par rotation du problème, mais s'avère moins réaliste pour représenter des robots mobiles (par exemple) que de prendre des agents *orientés*, ayant donc un "devant", et disposant d'actions telles que `avancer`, `tourner à droite` ou `tourner à gauche`.

Persistance de l'attention

En analysant les résultats de nos travaux dans le chapitre 8, on a évoqué l'intérêt que pourrait avoir l'ajouter d'un phénomène de *persistance de l'attention* (voir problème illustré par la figure 8.4). Or, si les agents utilisés étaient orientés, un tel phénomène serait naturel. En effet, une fois l'agent dirigé vers un objectif, celui-ci est largement favorisé. L'avantage d'un “agent orienté” est qu'il devrait ainsi moins hésiter même quand il combine des comportements (voir figure A.3).



a) La combinaison amène moins à hésiter.
Quand on combine deux comportements (figure a), l'agent hésite peu : aller de l'avant est plus efficace que se retourner.



b) Résultat après apprentissage.
Quand on apprend un unique comportement simple (figure b), l'ordre dans lequel sont perçus les objets est moins important : si l'agent est tourné vers le nord, cela reste sa direction privilégiée.

FIG. A.3 – Phénomène de persistance de l'attention dans le cas d'un agent “orienté”.

A.3.3 Sur les mondes toriques

Un dernier point que nous évoquerons, plus anecdotique que les précédents, est le choix d'environnements toriques (proie-prédateurs) ou non (fusion de blocs, monde des tuiles).

Cas torique

Si peu de problèmes réels se situent dans un environnement torique, le problème proie-prédateur est souvent considéré dans une grille dont les bords opposés se rejoignent. L'intérêt de ce choix est de ne pas tomber dans des situations particulières où la proie est prise entre prédateurs et mur. On préfère que les mouvements de tous les agents soient libres, les seuls obstacles étant constitués par les autres agents.

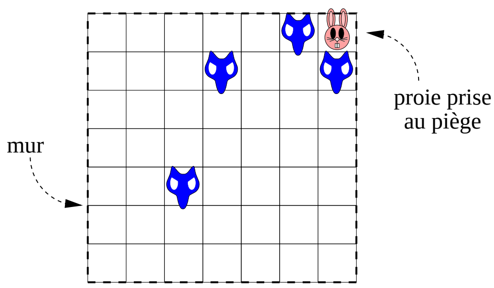
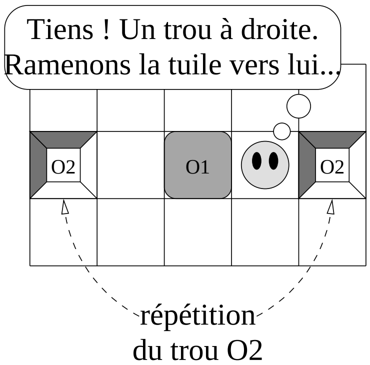


FIG. A.4 – Situation gênante quand le problème proie-prédateur est pris dans un monde clos.

Cas non torique

Dans le problème de fusion de blocs comme dans le monde des tuiles, le fait d'être dans un monde non torique a amené à interdire la présence de blocs ou de tuiles sur les cases le long du bord du terrain, de manière à ce qu'elles restent contrôlables (qu'on puisse les contourner pour aller les déplacer).

Pour expliquer notre choix d'utiliser des environnements non toriques, prenons comme exemple le monde des tuiles, avec un seul trou et une seule tuile. L'agent doit parfois aller faire le tour de la tuile qu'il perçoit pour la ramener vers le trou choisi. Or, ce faisant, le trou visé peut être perçu de manière complètement différente (voir figure A.5), ce qui va inciter l'agent à revenir sur ces pas, commençant ainsi un mouvement oscillatoire.



L'agent pourrait voir le trou comme étant à l'est ou à l'ouest. Mais les perceptions choisie font que la direction privilégiée est celle où le trou est le plus proche (ici à l'est).

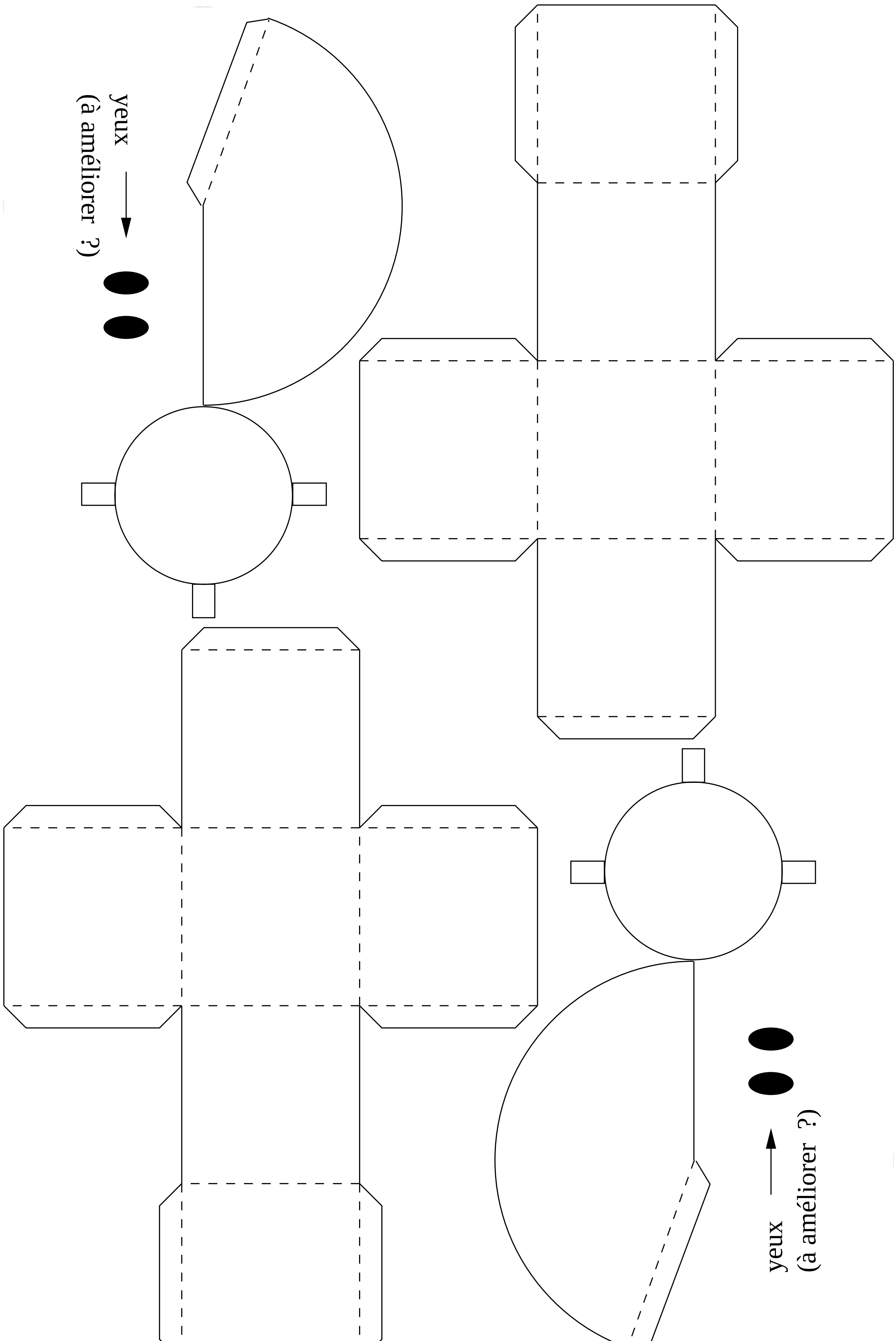
FIG. A.5 – Cas d'oscillation dû à l'aspect torique de l'environnement.

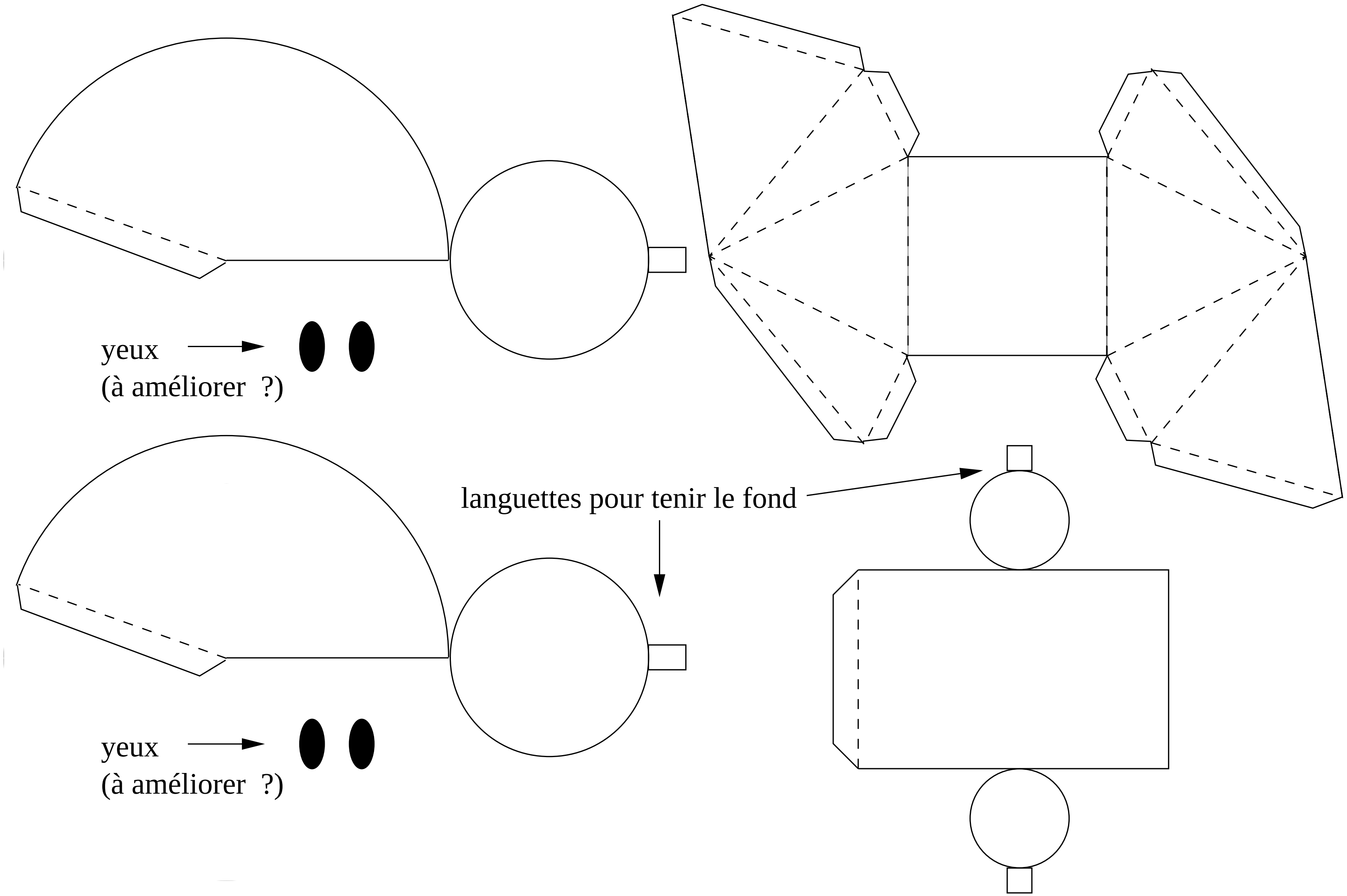
Annexe B

Découpages

B.1 Objets

B.2 Logo Maia


yeux
(à améliorer ?)
yeux
(à améliorer ?)


yeux
(à améliorer ?)
yeux
(à améliorer ?)
languettes pour tenir le fond

Publications

Conférences internationales avec comité de lecture et actes

[1] O. Buffet, A. Dutech, and F. Charpillet. Incremental reinforcement learning for designing multi-agent systems. In *Proceedings of the fifth international conference on Autonomous agents (Agents'01)*, 2001. [poster].

[2] A. Dutech, O. Buffet, and F. Charpillet. Multi-agent systems by incremental gradient reinforcement learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, 2001.

[3] O. Buffet, A. Dutech, and F. Charpillet. Learning to weigh basic behaviors in scalable agents. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'02)*, 2002. [poster].

[4] O. Buffet, A. Dutech, and F. Charpillet. Adaptive combination of behaviors in an agent. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'02)*, 2002.

[5] O. Buffet, A. Dutech, and F. Charpillet. Automatic generation of an agent's basic behaviors. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'03)*, 2003. [à paraître].

Colloques internationaux avec comité de lecture et actes

[6] O. Buffet and A. Dutech. Looking for scalable agents. In *Proceedings of the fifth European Workshop on Reinforcement Learning (EWRL-5)*, 2001.

Autres publications, rapports techniques et mémoires

[7] O. Buffet. Apprentissage par renforcement dans un système multi-agents. Mémoire de DEA, Université Henri Poincaré (Nancy), 2000.

[8] O. Buffet. Apprentissage par renforcement pour la conception de systèmes multi-agents. [rapport d'avancement de thèse], Laboratoire lorrain de recherche en informatique et ses applications (LORIA), 2002.

Bibliographie

[Arfken, 1985a] G. Arfken. *Gradient* ∇, chapitre 1.6, pages 33–37. Academic Press, 1985.

[Arfken, 1985b] G. Arfken. *The Method of Steepest Descents*, chapitre 7.4, pages 428–436. Academic Press, 1985.

[Arfken, 1985c] G. Arfken. *Successive Applications of* ∇, chapitre 1.9, pages 47–51. Academic Press, 1985.

[Asada et al, 1996] M. Asada et al. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23 :279–303, 1996.

[Atkinson *et al.*, 1996] R.L. Atkinson, R.C. Atkinson, E.E. Smith, D.J. Bem, et S. Nolen-Hoeksema. *Hilgard's Introduction to Psychology.* Harcourt Brace College Publishers, 1996. 12th edition.

[Aylett, 1995] R. Aylett. Multi-agent planning : Modelling execution agents. In *Papers of the 14th Workshop of the UK Planning and Scheduling Special Interest Group*, 1995.

[Baird et Moore, 1999] L.C. Baird et A.W. Moore. Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems 11.* The MIT Press, 1999.

[Baird, 1999] L.C. Baird. *Reinforcement Learning Through Gradient Descent.* Thèse de doctorat, Carnegie Mellon University, Pittsburgh, PA 15213, 1999.

[Bartlett et Baxter, 1999] P.L. Bartlett et J. Baxter. Hebbian synaptic modifications in spiking neurons that learn. Rapport technique, Australian National University, november 1999.

[Barto *et al.*, 1991] A.G. Barto, S. Bradtke, et S. Singh. Real-time learning and control using asynchronous dynamic programming. Rapport technique COINS 91-57, Dept. of Computer Science, University of Massachusetts, Amherst, 1991.

[Bass, 1992] T. Bass. Road to ruin. In *Discover*, pages 56–61. 1992.

[Baxter *et al.*, 2001] J. Baxter, P. Bartlett, et Lex Weaver. Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15 :351–381, 2001.

[Baxter et Bartlett, 2001] J. Baxter et P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15 :319–350, 2001.

[Bellman, 1957] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New-Jersey, 1957.

[Bellman, 1978] R. Bellman. *An Introduction to Artificial Intelligence : Can Computers Think ?* Boyd & Fraser Publishing Company, San Francisco, 1978.

[Benda *et al.*, 1986] M. Benda, V. Jagannathan, et R. Dodhiawala. On optimal cooperation of knowledge sources - and empirical investigation. Rapport technique, Boeing Advanced Technology Center, July 1986.

[Bernstein *et al.*, 2000] D.S. Bernstein, S. Zilberstein, et N. Immerman. The complexity of decentralized control of markov decision processes. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI'00), Stanford, California*, 2000.

[Bernstein *et al.*, 2002] D.S. Bernstein, R. Givan, N. Immerman, et S. Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4) :819–840, 2002.

[Bertsekas et Tsitsiklis, 1989] D.P. Bertsekas et J.N. Tsitsiklis. Parallel distributed computation. In *Numerical methods*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Bertsekas, 1987] D.P. Bertsekas. *Dynamic Programming : Deterministic and Stochastic Models*. Englewood Cliffs, NJ : Prentice-Hall, 1987.

[Bettelheim, 1976] Bruno Bettelheim. *Psychanalyse des contes de fées*. Robert Laffont, 1976.

[Birattari *et al.*, 2002] M. Birattari, G. Di Caro, et M. Dorigo. Toward the formal foundation of ant programming. In *Ant Algorithms. Third International workshop (ANTS 2002), Brussels, Belgium*, 2002.

[Blumberg, 1994] B. Blumberg. Action-selection in hamsterdam : Lessons from ethology. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB'94)*, 1994.

[Bonabeau *et al.*, 1999] E. Bonabeau, M. Dorigo, et G. Theraulaz. *Swarm Intelligence : From Natural to Artificial Systems*. Oxford University Press, New-York, NY, 1999.

[Bonabeau et Theraulaz, 1994] E. Bonabeau et G. Theraulaz. *Intelligence Collective*. Hermès, Paris, 1994.

[Bond et Gasser, 1988] A.H. Bond et L. Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufman Publishers : SanMateo, CA, 1988.

[Bourjot *et al.*, 2003] C. Bourjot, V. Chevrier, et V. Thomas. A new swarm mechanism based on social spiders colonies : from web weaving to region detection. *Web Intelligence and Agent Systems : An International Journal - WIAS*, Mars 2003.

[Boutilier, 1996] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '96), De Zeeuwse Stromen, The Netherlands*, 1996.

[Boutilier, 1999] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99), Stockholm, Sweden*, 1999.

[Bouzid, 2001] Makram Bouzid. *Contribution à la modélisation de l'interaction agent/environnement : Modélisation stochastique et simulation parallèle*. Thèse d'université, Université Henri Poincaré - Nancy 1, Novembre 2001.

[Brassac et Pesty, 1995] C. Brassac et S. Pesty. Coopération dans les systèmes multi-agents, comportement ou conduite ? In *Proceedings of the first International Workshop on Decentralized Intelligent and MultiAgent Systems, Krakow, Poland*, 1995.

[Brooks et Stein, 1994] R.A. Brooks et L.A. Stein. Building brain for bodies. Rapport technique, M.I.T., August 1994. A.I. Memo No.1439.

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

[Brooks, 1989] R.A. Brooks. A robot that walks ; emergent behavior from a carefully evolved network. *Neural Computation*, 1(2) :253–262, 1989.

[Brooks, 1991a] R.A. Brooks. Intelligence without reason. In *Proceedings of the Twelth International Joint Conference on Artificial Intelligence (IJCAI'91), Sydney, Australia*, 1991.

[Brooks, 1991b] R.A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47 :139–159, 1991.

[Buffet, 2000] O. Buffet. Apprentissage par renforcement dans un système multi-agents. Mémoire de maîtrise, Université Henri Poincaré (Nancy), 2000. Mémoire de DEA.

[Cassandra, 1998] A. R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. Thèse de doctorat, Brown University, Department of Computer Science, Providence, RI, 1998.

[Chadès, 2003] I. Chadès. *Planification Distribuée dans les Systèmes Multi-agents à l'aide de Processus Décisionnels de Markov*. Thèse de doctorat, Université Henri Poincaré, Nancy 1, 2003.

[Chaib-draa, 1999] B. Chaib-draa. *Agents et Systèmes Multi-Agents. Notes de Cours*. Notes de Cours, Département informatique, Faculté des Sciences et de Génie, Université de Laval, Québec, 1999. http ://www.damas.ift.ulaval.ca/~chaib/cours.pdf.

[Charniak et McDermott, 1985] E. Charniak et D. McDermott. *Introduction to Artificial Intelligence*. Addison-Wesley, Reading, Massachusetts, 1985.

[Clancey, 1997] W. J. Clancey. *Situated Cognition*. Cambridge University Press, Cambridge., 1997.

[Colombetti *et al.*, 1996] M. Colombetti, M. Dorigo, et G. Borghi. Robot shaping : The hamster experiment. In *Proceedings of the Sixth International Symposium on Robotics and Manufacturing (ISRAM'96)*, Montpellier, France, 1996.

[Connell et Mahadevan, 1993] J. Connell et S. Mahadevan. *Rapid Task Learning for Real Robots*. Kluwer Academic publishers, 1993.

[Demazeau, 1997] Y. Demazeau. Steps towards multi-agent oriented programming. In *First International Workshop on Multi-Agent Systems (IWMAS'97)*, 1997.

[Deneubourg *et al.*, 1990] J.-L. Deneubourg, S. Aron, S. Goss, et J.-M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3 :159–168, 1990.

[Dietterich, 2000] T.G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13 :227–303, 2000.

[Dixon *et al.*, 2000] K. Dixon, R. Malak, et P. Khosla. Incorporating prior knowledge and previously learned information into reinforcement learning agents. Rapport technique, Carnegie Mellon University, Institute for Complex Engineered Systems, 2000.

[Dorigo *et al.*, 1991] M. Dorigo, V. Maniezzo, et A. Colorni. Positive feedback as a search strategy. Rapport technique, Dipartimento di Elettronica e Informatica, Politecnico di Milano, IT, 1991. 91016.

[Dorigo et Colombetti, 1997] M. Dorigo et M. Colombetti. Book precis : Robot shaping : An experiment in behaviour engineering. *Adaptive Behavior*, 5(3-4) :391–406, 1997. In Special issue on Complete Agent Learning in Complex Environments.

[Dorigo et Di Caro, 1999] M. Dorigo et G. Di Caro. Ant colony optimization : A new meta-heuristic. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[Drummond, 2002] C. Drummond. Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of Artificial Intelligence Research*, 16 :59–104, 2002.

[Dubois *et al.*, 1994] D. Dubois, J. Lang, et H. Prade. Possibilistic logic. In Dov Gabbay, Christopher J. Hogger, et J. A. Robinson, éditeurs, *Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3 : Nonmonotonic Reasoning and Uncertain Reasoning*, pages 439–513. Oxford University Press, Oxford, 1994.

[Dury, 2000] A. Dury. *Modélisation des interactions dans les systèmes multi-agents*. Thèse de doctorat, LORIA, Nancy, 2000.

[Dutech *et al.*, 2001] A. Dutech, O. Buffet, et F. Charpillet. Multi-agent systems by incremental gradient reinforcement learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, 2001.

[Dutech et Samuelides, 2003] A. Dutech et M. Samuelides. Un algorithme d'apprentissage par renforcement pour les processus decisionnels de markov partiellement observes : apprendre une rextension selective du passé. *Revue d'Intelligence Artificielle*, à paraître, 2003.

[Dutech, 1999] A. Dutech. *Apprentissage d'environnement : approches cognitives et comportementales*. Thèse de doctorat, ENSAE, Toulouse, 1999.

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

[Ferber et Müller, 1996] J. Ferber et J.-P. Müller. Influences and reaction : a model of situated multiagent systems. In *Proceedings of the 2^{nd} International Conference on Multi-Agent Systems (ICMAS'96)*, 1996.

[Ferber, 1995] J. Ferber. *Les Systèmes multi-agents*. InterEditions, 1995.

[Ferguson, 1992] I.A. Ferguson. *TouringMachines : An Architecture for Dynamic, Rational, Mobile Agents*. Thèse de doctorat, University of Cambridge, UK, 1992.

[Fox, 1981] M. S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1) :70–80, January 1981.

[Gadanho et Custodio, 2002] S. Gadanho et L. Custodio. Asynchronous learning by emotions and cognition. In *Proceedings of the Seventh International Conference on the Simulation of Adaptive Behavior (SAB'02)*, 2002.

[Gardner, 1970] Martin Gardner. Mathematical games : The fantastic combinations of john conway's new solitaire game 'life'. *Scientific American*, pages 120–123, October 1970.

[Grassé, 1959] P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis* et *cubitermes sp.*. la théorie de la stigmergie : Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6 :41–81, 1959.

[Harnad, 1990] S. Harnad. The symbol grounding problem. *Physica D*, 42 :335–346, 1990.

[Haugeland, 1985] J. Haugeland. *Artificial Intelligence : The Very Idea*. MIT Press, Cambridge, Massachusetts, 1985.

[Hayes-Roth, 1985] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 3(26) :251–321, 1985.

[Hebb, 1949] D. O. Hebb. *The organisation of behavior*. Wiley, 1949.

[Hopfield, 1982] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79 :2554–2558, 1982.

[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.

[Hu et Wellman, 1998a] J. Hu et M. Wellman. Multiagent reinforcement learning : theoretical framework and an algorithm. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, pages 242–250, 1998.

[Hu et Wellman, 1998b] J. Hu et M. Wellman. Online learning about other agents in a dynamic multiagent system. In *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 239–246, 1998.

[Humphrys, 1996] M. Humphrys. Action selection methods using reinforcement learning. In *From Animals to Animats 4 : 4th International Conference on Simulation of Adaptive Behavior (SAB-96)*, September 1996.

[Humphrys, 1997] M. Humphrys. *Action Selection methods using Reinforcement Learning*. Thèse de doctorat, University of Cambridge, 1997.

[Jaakkola *et al.*, 1994a] T. Jaakkola, M. Jordan, et S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6) :1186–1201, 1994.

[Jaakkola *et al.*, 1994b] T. Jaakkola, M. Jordan, et S. Singh. Reinforcement learning algorithm for partially observable markov decision problems. In G. Tesauro, D. Touretzky, et T. Leen, éditeurs, *Advances in Neural Information Processing Systems*, volume 7, pages 345–352. The MIT Press, 1994.

[Jennings *et al.*, 1998] N.R. Jennings, K. Sycara, et M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1 :7–38, 1998.

[Jørgensen, 1962] J. Jørgensen. *Psykologi – paa biologisk Grundlag*. Scandinavian University Books, Munksgaard, København, 1962.

[Joslin *et al.*, 1993] D. Joslin, A. Nunes, et M. E. Pollack. Tileworld users' manual. Rapport technique TR 93-12, August 1993.

[Kaelbling *et al.*, 1996] L. Kaelbling, M. Littman, et A. Moore. Reinforcement learning : A survey. *Journal of Artificial Intelligence Research*, 4 :237–285, 1996.

[Kurzweil, 1990] R. Kurzweil. *The Age of Intelligent Machines*. MIT Press, Cambridge, Massachusetts, 1990.

[Laurent, 2002] G. Laurent. *Synthèse de comportements par apprentissages par renforcement parallèles*. Thèse de doctorat, Université de Franche-Comté, 2002.

[Lin, 1993] L.-J. Lin. Scaling up reinforcement learning for robot control. In *Proceedings of the Tenth International Conference on Machine Learning (ICML'93)*, 1993.

[Littman et R. Sutton, 2001] M. Littman et S. Singh R. Sutton. Predictive representations of state. In *Advances in Neural Information Processing Systems 14 (NIPS'01)*, 2001.

[Littman, 1994a] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML'94), San Fransisco, CA*, 1994.

[Littman, 1994b] M. L. Littman. The witness algorithm : Solving partially observable markov decision processes. Rapport technique CS-94-40, Brown University, Department of Computer Science, Providence, Rhode Island, 1994.

[Lloyd, 1833] W. F. Lloyd. *Two Lectures on the Checks to Population*. Oxford University Press, Oxford, England, 1833.

[Loch et Singh, 1998] Loch et S. Singh. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*, 1998.

[Luger et Stubblefield, 1993] G.F. Luger et W.A. Stubblefield. *Artificial Intelligence : Structures and Strategies for Complex Problem Solving.* Benjamin/Cummings, Redwood City, California, second edition, 1993.

[Madani *et al.*, 1999] Omid Madani, Steve Hanks, et Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *Sixteenth National Conference on Artificial Intelligence AAAI/IAAI*, 1999.

[Maes, 1989] P. Maes. How to do the right thing. *Connection Science*, 1 :291–323, 1989.

[Mahadevan, 1996] Sridhar Mahadevan. An average-reward reinforcement learning algorithm for computing bias-optimal policies. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI '96)*, 1996.

[Malville, 1999] E. Malville. *L'auto-organisation de groupes pour l'allocation de taches dans les Systèmes Multi-Agents : Application à Corba.* Thèse de doctorat, Université de Savoie, 1999.

[Matarić, 1994] Maja J Matarić. Reward functions for accelerated learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 181–189. Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[Matarić, 1997] M. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1) :73–83, 1997.

[McCallum, 1995] R. A. McCallum. *Reinforcement Learning with Selective Perception and Hidden State.* Thèse de doctorat, University of Rochester, 1995.

[McGovern *et al.*, 1998] A. McGovern, D. Precup, B. Ravindran, S. Singh, et R. Sutton. Hierarchical optimal control of mdps. In *Proceedings of the 10th Yale Workshop on Adaptive and Learning systems*, 1998.

[Meuleau *et al.*, 1998] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L.P. Kaelbling, T. Dean, et C. Boutilier. Solving very large weakly coupled markov decision processes. In *AAAI/IAAI*, pages 165–172, 1998.

[Meuleau *et al.*, 1999] N. Meuleau, L. Peshkin, K.-E. Kim, et L.P. Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI'99)*, pages 427–436, 1999.

[Meuleau et Dorigo, 2002] N. Meuleau et M. Dorigo. Ant colony optimization and stochastic gradient descent. *Artificial Life*, 8(2), 2002.

[Minsky, 1986] M. Minsky. *La Société de l'Esprit.* InterEditions, 1986.

[Monahan, 1982] G. Monahan. A survey of partially observable markov decision processes. *Management Science*, 28 :1–16, 1982.

[Moore, 1990] A. Moore. *Efficient memory-based learning for robot control.* Thèse de doctorat, Trinity Hall, University of Cambridge, 1990.

[Moore, 1991] A. Moore. Variable resolution dynamic programming : Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of the 8th International Conference on Machine Learning (ICML'91).* Morgan Kaufmann, 1991.

[Müller et Pischel, 1993] J. Müller et M. Pischel. The agent architecture interrap : Concept and application. Rapport technique, DFKI Saarbrucken, 1993. RR-93-26.

[Müller, 1997] J. Müller. Control architectures for autonomous and interacting agents : A survey. In L. Cavedon, A. Rao, et W. Wobcke, éditeurs, *Intelligent Agents Systems : Theoretical and Practical Issues, Lecture Notes in A.I.* Springer, 1997.

[Munos et Moore, 2002] R. Munos et A. Moore. Variable resolution discretization in optimal control. *Machine Learning*, 2002.

[Munos, 1997] R. Munos. *Apprentissage par renforcement, étude du cas continu.* Thèse de doctorat, Ecole des Hautes études en Sciences Sociales, 1997.

[Neumann et Morgenstern, 1967] J. Von Neumann et O. Morgenstern. *Theory of games and economic behavior.* Princeton University Press, 1967.

[Ng *et al.*, 1999] A.Y. Ng, D. Harada, et S. Russell. Policy invariance under reward transformations : Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML'99)*, 1999.

[Owen, 1982] G. Owen. *Game Theory : Second Edition.* Academic Press, Orlando, Florida, 1982.

[Peshkin *et al.*, 1999] L. Peshkin, N. Meuleau, et L.P. Kaelbling. Learning policies with external memory. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML'99)*, pages 307–314, 1999.

[Peshkin *et al.*, 2000] L. Peshkin, K. Kim, N. Meuleau, et L. Kaelbling. Learning to cooperate via policy search. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI'00)*, 2000.

[Pfeifer, 2000] R. Pfeifer. the role of morphology and materials in adaptive behavior. In *From Animals to Animats 6 : Proceedings of the 6th International Conference on Simulation of Adaptive Behavior (SAB-00)*, 2000.

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

[Precup et Sutton, 97] D. Precup et R. Sutton. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems 10 (NIPS'97)*, 97.

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

[Randløv et Alstrøm, 1998] J. Randløv et P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the 15th International Conference on Machine Learning, (ICML-98)*, pages 463–471, 1998.

[Randløv, 2000] J. Randløv. Shaping in reinforcement learning by changing the physics of the problem. In *Proceedings of the 17th International Conference on Machine Learning, (ICML-00)*, pages 767–774, 2000.

[Rescorla et Wagner, 1972] S.A. Rescorla et A.R. Wagner. A theory of pavlovian conditioning : variations in effectiveness of reinforcement and nonreinforcement. In *Classical Conditioning*, volume II, pages 64–99. New York : Appleton-Century-Crofts, 1972.

[Rich et Knight, 1991] E. Rich et K. Knight. *Artificial Intelligence.* McGraw-Hill, New-York, second edition, 1991.

[Rosenblatt, 1995] J. Rosenblatt. Damn : A distributed architecture for mobile navigation. In *Proceedings of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, pages 27–29, 1995.

[Russell et Norvig, 1995] S. Russell et P. Norvig. *Artificial Intelligence : A Modern Approach.* Englewood Cliffs, NJ : prentice Hall, 1995.

[Sahota, 1994] M.K. Sahota. Action selection for robots in dynamic environments through inter-behaviour bidding. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB'94)*, 1994.

[Schalkoff, 1990] R.J. Schalkoff. *Artificial Intelligence : An Engineering Approach.* McGraw-Hill, New-York, 1990.

[Scherrer, 2003] B. Scherrer. *Apprentissage de représentation et auto-organisation modulaire pour un agent autonome.* Thèse de doctorat, Université Henri Poincaré, Nancy 1, 2003.

[Shapley, 1953] L. S. Shapley. Stochastic games. *National Academy of Sciences of the United States of America*, 39 :1095–1100, 1953.

[Shapley, 1964] L. S. Shapley. Some topics in two person games. *Annals of Mathematical Studies*, 5 :1–28, 1964.

[Singh *et al.*, 1994] S. Singh, T. Jaakkola, et M. Jordan. Learning without state estimation in partially observable markovian decision processes. In *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*, 1994.

[Singh et Cohn, 1998] S. Singh et D. Cohn. How to dynamically merge markov decision processes. In *Advances in Neural Information Processing Systems 10 (NIPS'98)*, 1998.

[Singh, 1992] S.P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8 :323–339, 1992.

[Skinner, 1938] B.F. Skinner. *The Behavior of Organisms : An Experimental Analysis.* Prentice Hall, Englewood Cliffs, New-Jersey, 1938.

[Skinner, 1953] B.F. Skinner. *Science and Human Behavior.* Collier-Macmillian, New-York, 1953.

[Smallwood et Sondik, 1973] R.D. Smallwood et E.J. Sondik. The optimal control of partially observable markovdecision processes over a finite horizon. *Operation Research*, 21 :1071–1088, 1973.

[Smith, 1980] Reid Smith. The contract net protocol : High level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12) :1104–1113, December 1980.

[Staddon, 1983] J.E.R. Staddon. *Adaptative Behavior and Learning.* Cambridge University Press, 1983.

[Stone et Veloso, 2000] P. Stone et M. Veloso. Multiagent systems : a survey from a machine learning perspective. *Autonomous Robotics*, 8(3), 2000.

[Sutton *et al.*, 1998] R. Sutton, D. Precup, et S. Singh. Between MDPs and Semi-MDPs : Learning, planning, and representing knowledge at multiple temporal scales. Rapport technique, University of Massachusetts, Department of Computer and Information Sciences, Amherst, MA, 1998.

[Sutton et Barto, 1998] R. Sutton et G. Barto. *Reinforcement Learning : an introduction.* Bradford Book, MIT Press, Cambridge, MA, 1998.

[Sutton, 1990] R. Sutton. Integrated architecture for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning (ICML'90)*, pages 216–224, 1990.

[Tesauro, 1992] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, (8) :257–277, 1992.

[Tesauro, 1994] G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. Rapport technique, IBM, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, 1994.

[Tham et Prager, 1994] C.K. Tham et R.W. Prager. A modular q-learning architecture for manipulator task decomposition. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, 1994.

[Thiébaux *et al.*, 2002] S. Thiébaux, F. Kabanza, et J. Slaney. Anytime state-based solution methods for decision processes with non-markovian rewards. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI'02)*, pages 501–510, July 2002.

[Thomas *et al.*, 2002] V. Thomas, C. Bourjot, V. Chevrier, et D. Desor. MAS and RATS : Multi-agent simulation of social differentiation in rats' groups. . In *International Workshop on Self-Organization and Evolution of Social Behaviour, Monte Verita, Ascona, Switzerland*, 2002.

[Thrun, 1992] S. Thrun. Efficient exploration in reinforcement learning. Rapport technique CMU-CS-92-102, Carnegie Mellon University, 1992.

[Tumer et Wolpert, 2000] K. Tumer et D. Wolpert. Collective intelligence and braess' paradox. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence, Austin, TX*, 2000.

[Tyrrell, 1993] T. Tyrrell. *Computational Mechanisms for Action Selection.* Thèse de doctorat, University of Edinburgh, 1993.

[Vera et Simon, 1993] A. H. Vera et H. A. Simon. Situated action : A symbolic interpretation. *Cognitive Science*, 17 :7–48, 1993.

[Watkins et Dayan, 1992] C. Watkins et P. Dayan. *q*-learning. *Machine Learning*, 8 :279–292, 1992.

[Watkins, 1989] C.J.C.H. Watkins. *Learning from delayed rewards.* Thèse de doctorat, King's College of Cambridge, UK., 1989.

[Weiß et Sen, 1996] G. Weiß et S. Sen. *Adaptation and learning in multi-agent systems*, volume 1042. Springer-Verlag, Lectures Notes in Artificial Intelligence, 1996.

[Weng *et al.*, 2002] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, et E. Thelen. Autonomous mental development by robots and animals. *Science Magazine*, 291(5504) :599–600, january 2002.

[Weng, 2002] J. Weng. A theory of mentally developing robots. In *Proceedings of the 2nd International Conference on Development and Learning (ICDL'02), MIT, Cambridge, MA*, june 2002.

[Williams et Baird, 1993] R. Williams et L. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Rapport technique NU-CCS-93-14, Northeastern University, november 1993.

[Williams, 1987] R.J. Williams. A class of gradient-estimating algorithms for reinforcement learning in neural networks. In *Proceedings of the 1st International Conference on Neural Networks (ICNN'87), San Diego, California*, 1987.

[Williams, 1992] R.J. Williams. Simple statistical gradient-following algorithms for connectionnist reinforcement learning. *Machine Learning*, 8(3) :229–256, 1992.

[Wilson, 1991] S.W. Wilson. The animat path to ai. In *From Animals to Animats : Proceedings of The First International Conference on Simulation of Adaptive Behavior*, pages 15–21, 1991.

[Winston, 1992] P.H. Winston. *Artificial Intelligence.* Addison Wesley, Reading, Massachusetts, third edition, 1992.

[Wolisz et Tschammer, 1993] A. Wolisz et V. Tschammer. Performance aspects of trading in open distributed systems. *Computer Communications*, 16 :277–287, May 1993.

[Wolpert *et al.*, 1999] D. Wolpert, K. Wheeler, et K. Tumer. General principles of learning-based multi-agent systems. In *Proceedings of the 3rd International Conference on Autonomous Agents (Agents'99), Seattle*, pages 77–83, 1999.

[Wooldridge *et al.*, 1995] M. Wooldridge, J.-P. Müller, et M. Tambe. Agent theories, architectures and languages : A bibliography. In *Intelligent Agents II, IJCAI'95 Workshop*, pages 408–431, 1995.

[Zadeh, 1973] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3(1) :28–44, 1973.

Index

Résumé

Cette thèse s'est intéressée à deux domaines de l'intelligence artificielle : d'une part l'apprentissage par renforcement (A/R), et d'autre part les systèmes multi-agents (SMA). Le premier permet de concevoir des agents (entités intelligentes) en se basant sur un signal de renforcement qui récompense les décisions menant au but fixé, alors que le second concerne l'intelligence qui peut venir de l'interaction d'un groupe d'entités (dans la perspective que le tout soit plus que la somme de ses parties).

Chacun de ces deux outils souffre de diverses difficultés d'emploi. Le travail que nous avons mené a permis de montrer comment chacun des deux outils peut servir à l'autre pour répondre à certains de ces problèmes. On a ainsi conçu les agents d'un SMA par A/R, et organisé l'architecture d'un agent apprenant par renforcement sous la forme d'un SMA. Ces deux outils se sont avérés très complémentaires, et notre approche globale d'une conception "progressive" a prouvé son efficacité.

Mots-clés: apprentissage par renforcement, systèmes multi-agents, processus de décision markoviens, sélection d'action, apprentissage progressif

Abstract

A Twofold Modular Approach of Reinforcement Learning for Adaptive Intelligent Agents

This PhD thesis has been interested in two fields of artificial intelligence : reinforcement learning (RL) on the one hand, and multi-agent systems (MAS) on the other hand. The former allows for the conception of agents (intelligent entities) based on a reinforcement signal which rewards decisions leading to the specified goal, whereas the latter is concerned with the intelligence that can result from the interaction of a group of entities (in the perspective that the whole is more than the sum of its parts).

Both these tools suffer from various difficulties. The work we accomplished has shown how these tools can serve each other to answer some of these problems. Thus, agents of a MAS have been conceived through RL, and the architecture of a reinforcement learning agent has been designed as a MAS. Both tools appear to be very complementary, and our global approach of a "progressive" design has proved its efficiency.

Keywords: reinforcement learning, multi-agent systems, Markov decision processes, action selection, shaping

Thèse de doctorat en informatique effectuée au sein de l'équipe MAIA du Loria.