Développement autonome des comportements de base d'un agent

Olivier Buffet[*,**] — **Alain Dutech**[**] — **François Charpillet**[**]

[*] *National ICT Australia / The Australian National University*
RSISE Building 115 - ANU / Canberra ACT 0200 / Australia
olivier.buffet@nicta.com.au

[**] *Loria - INRIA-Lorraine / Campus Scientifique - BP 239*
54506 Vandœuvre-lès-Nancy cedex / France
{dutech,charp}@loria.fr

RÉSUMÉ. *La problématique abordée dans cet article est celle de la conception automatique d'agents autonomes devant résoudre des tâches complexes mettant en œuvre plusieurs objectifs potentiellement concurrents. Nous proposons alors une approche modulaire s'appuyant sur les principes de la sélection d'action où les actions recommandées par plusieurs comportements de base sont combinées en une décision globale. Dans ce cadre, notre principale contribution est une méthode pour qu'un agent puisse définir et construire automatiquement les comportements de base dont il a besoin* via *des méthodes d'apprentissage par renforcement incrémentales. Nous obtenons ainsi une architecture très autonome ne nécessitant que peu de réglages. Cette approche est testée et discutée sur un problème représentatif issu du "monde des tuiles".*

ABSTRACT. *The problem addressed in this article is that of automatically designing autonomous agents having to solve complex tasks involving several –and possibly concurrent– objectives. We propose a modular approach based on the principles of action selection where the actions recommanded by several basic behaviors are combined in a global decision. In this framework, our main contribution is a method making an agent able to automatically define and build the basic behaviors it needs through incremental reinforcement learning methods. This way, we obtain a very autonomous architecture requiring very few hand-coding. This approach is tested and discussed on a representative problem taken from the "tile-world".*

MOTS-CLÉS : *problèmes de décision markoviens, apprentissage par renforcement, motivations multiples.*

KEYWORDS: *Markov Decision Problems, Reinforcement Learning, Multiple Motivations.*

RSTI - RIA – 19/2005. Apprentissage automatique, pages 603 à 632

1. Introduction

Un agent (Russell *et al.,* 1995, Jennings *et al.,* 1998) est défini ici comme une entité immergée dans un environnement au sein duquel cette entité cherche à atteindre un objectif qui lui est propre. Dans ce but, l'agent interagit avec son environnement : il le perçoit et peut agir sur celui-ci, comme illustré sur la figure 1.

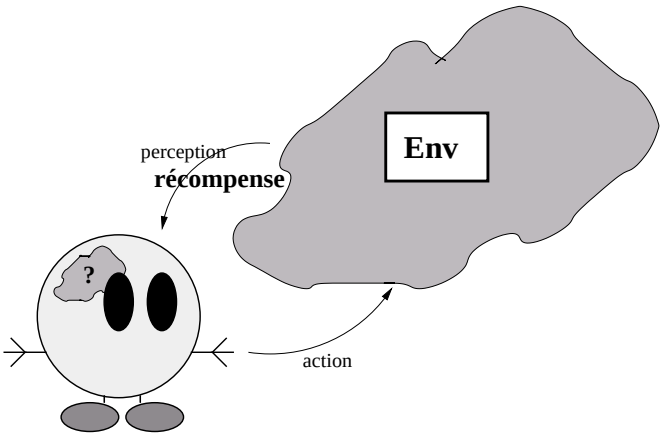


Figure 1. *Agent et son environnement liés à travers une boucle perception/action*

Dans nos travaux, nous cherchons à concevoir de manière automatisée des agents se trouvant dans des milieux à la fois inconnus, incertains et partiellement observés (l'état précis du système agent+environnement n'est pas connu). Sous ces hypothèses difficiles, nous nous limitons à considérer des agents réactifs, adoptant un comportement de type stimulus-action.

Notre intérêt se porte sur la synthèse de ce comportement par l'utilisation de méthodes d'*apprentissage par renforcement* (A/R). Dans ce cadre, l'objectif assigné à un agent est défini par des récompenses lui indiquant quand un événement considéré comme positif ou négatif survient. L'apprentissage par renforcement consiste pour un agent à renforcer les décisions qui lui permettent de maximiser les récompenses qu'il reçoit. Si de nombreuses recherches ont déjà été conduites sur cette thématique, nous nous intéressons ici plus particulièrement à la situation dans laquelle un agent possède plusieurs motivations éventuellement conflictuelles.

L'apprentissage par renforcement passe classiquement par le formalisme des problèmes de décision markoviens (MDP). Si ceux-ci peuvent *a priori* tenir compte de différentes motivations du moment qu'elles se traduisent par des fonctions de récompenses additives ($r = r_1 + r_2 + \dots$), il devient vite déraisonnable d'apprendre un comportement dans de grands espaces d'états. Dans l'hypothèse que l'environnement est constitué d'objets en nombre variable et que les motivations courantes de l'agent dépendent des objets présents, une approche possible est celle des MDP *relationnels* (Boutilier *et al.,* 2001, Dzeroski *et al.,* 2001, Gretton *et al.,* 2004) (voir figure 2). Mal-

heureusement, on ne peut aisément étendre cette approche dans le cas d'un agent avec observations partielles et sans mémoire à court terme (agent à comportement réactif), situation dans laquelle il faut chercher une politique stochastique, donc difficilement représentable par des outils de logique comme ceux utilisés par l'A/R relationnel.

Figure 2. *Les approches relationnelles pour l'apprentissage par renforcement permettent entre autres d'utiliser la même connaissance pour mettre le bloc* `[A]` *sur* `[C]` *dans le cas de gauche que pour mettre* `[C]` *sur* `[B]` *ou* `[D]` *(même couleurs) dans le cas de droite. Dans ce second cas, on a deux motivations concurrentes : mettre* `[C]` *sur* `[B]` *ou mettre* `[C]` *sur* `[D]`

1.1. *Approche : sélection d'action et apprentissage par renforcement*

L'apprentissage par renforcement relationnel classique répondant mal au problème de la prise de décision avec motivations multiples et en milieu partiellement observé (sans mémoire à court terme), nous avons orienté nos recherches vers des méthodes heuristiques de combinaison de comportements (on parle de sélection d'action (Tyrrell, 1993), voir section 2.3) pour voir comment elles peuvent entrer dans la conception d'un agent par apprentissage par renforcement.

Suivant les mêmes principes que (Humphrys, 1996), nous employons une architecture de sélection d'action dans laquelle un agent connaît différents comportements de bas niveau (que nous appelons *comportements de base*) et doit choisir une action d'après l'utilité/la préférence que lui associent ces comportements de base. De précédents travaux dans le domaine de l'A/R pour la sélection d'action se sont concentrés :

– soit sur l'adaptation du processus de combinaison de comportements de base (Buffet *et al.*, 2002),

– soit sur l'apprentissage/l'amélioration de comportements de base sélectionnés au préalables (Humphrys, 1996, Buffet *et al.*, 2003).

Alors que ces travaux proposent de fournir à l'agent des comportements sélectionnés manuellement, notre objectif est que l'agent puisse définir *de lui-même* les comportements de base dont il a besoin pour prendre de bonnes décisions, de manière :

– à compléter, voire remplacer des choix intuitifs de ces comportements, et

– à automatiser autant que possible la conception.

Pour cela, nous suggérons de donner à l'agent les moyens d'apprendre, partant de zéro, ces comportements. Ceci signifie non seulement apprendre à accomplir une tâche donnée, mais aussi et surtout *déterminer quelles tâches apprendre*.

Ce type d'apprentissage est rendu possible dans le cadre général de l'apprentissage par renforcement grâce à un nouveau mécanisme de sélection d'action (décrit et étudié en détails dans (Buffet *et al.*, 2003)) qui permet à un agent de combiner de manière adaptative des comportements de base en un comportement complexe efficace. Nous montrons que ce même mécanisme peut être utilisé par l'agent pour établir si le nouveau comportement appris, plus complexe, mérite d'être employé comme un *nouveau comportement de base dans le processus de sélection d'action futur*. Ainsi, sans connaissances préalables, notre agent fait croître de manière incrémentale son propre *ensemble* de comportements de base répondant à ses besoins et buts.

1.2. *Cadre de travail*

Afin d'éviter de possibles méprises, clarifions aussi dès à présent les bases de notre architecture d'agent :

– l'agent ne connaît pas de modèle de la dynamique de son environnement et ne l'observe que partiellement. Les buts de l'agent sont représentés par un signal de récompense scalaire dépendant de l'action et de l'état de l'agent ;

– l'agent a un ensemble de comportements de base, chacun répondant à un but/une motivation simple. En outre, chaque comportement de base est de type stimulus-réaction, guidé par la seule observation courante (pas de mémoire à court terme) ;

– dans une situation donnée (perception + choix d'un objectif), l'agent utilise l'information (principalement l'utilité des actions) de ses divers comportements de base pour décider de l'action à effectuer.

1.3. *Exemples*

Voici deux exemples de problèmes auxquels notre travail s'applique. Le premier est un problème applicatif (qui nécessiterait une mise en forme plus complète), et le second servira à valider l'approche proposée.

Assistant personnel intelligent

Un certain nombre de projets actuels visent à rendre notre environnement (maison, ville, entreprise) "intelligent". Un outil privilégié dans ce cadre est l'assistant personnel (PDA), outil informatique portatif que l'on va doter ici de la capacité de communiquer avec d'autres appareils l'environnant.

Au sein d'une maison, l'assistant pourrait être capable d'accomplir des tâches en combinant les capacités de différents appareils (afficher sur la télévision une image stockée dans un appareil photo numérique par exemple). Mais les appareils dispo-

nibles ne sont pas toujours les mêmes d'un instant à l'autre, et l'assistant peut avoir à gérer différentes demandes à la fois.

On se retrouve donc dans une situation avec plusieurs motivations simultanées, avec des objets (appareils) apparaissant/disparaissant de son environnement (selon qu'ils sont utilisés, déplacés...). Cela correspond précisément à la problématique que nous avons définie précédemment.

Le monde des tuiles

Pour illustrer notre approche et tester nos algorithmes, nous emploierons le classique problème du monde des tuiles (voir (Joslin *et al.*, 1993)). Dans celui-ci, tel que représenté sur la figure 3, l'agent doit pousser des tuiles dans des trous tout en évitant d'y tomber lui-même. Intuitivement, il faut pour cela deux comportements de base : pousser des tuiles dans des trous, et éviter des trous.

Avec notre algorithme d'apprentissage, un agent devrait apprendre seul quels comportements de base sont réellement utiles pour résoudre cette tâche complexe.

Dans ce cadre, le monde des tuiles est des plus intéressants car :

– il fait apparaître des comportements aussi bien positifs (essayer de pousser un tuile dans un trou) que négatifs (éviter les trous) ;

– l'un des comportements implique de manipuler plus d'un objet, ce qui évite de se limiter à des interactions trop simples ;

– des objets peuvent avoir plusieurs "rôles" : une tuile peut n'être qu'un obstacle dans certaines situations, ou pourrait être placée dans plusieurs trous disponibles.

Nous verrons tout cela à travers les expérimentations.

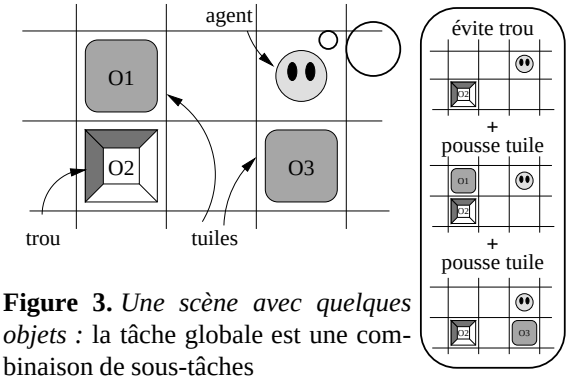


Figure 3. *Une scène avec quelques objets :* la tâche globale est une combinaison de sous-tâches

1.4. *Organisation de l'article*

Pour présenter notre algorithme de définition automatique de comportements de base, nous commençons par définir de manière plus précise la notion de *motivation*

et revenons sur les outils que sont l'apprentissage par renforcement et la sélection d'action (section 2). Ensuite la section 3 introduit la notion de *comportement de base*, avant d'expliquer comment des comportements de base sont combinés en de nouveaux comportements, ces derniers étant des nouveaux comportements de base potentiels. Le processus d'apprentissage d'un ensemble de comportements de base lui-même est présenté dans la section 4, expérimenté et analysé en section 5. Une discussion sur notre algorithme et quelques travaux futurs conclut cet article.

2. Préliminaires

Après avoir introduit quelques notations utiles, cette section présente la notion de motivation, puis revient sur les principaux outils employés dans notre approche : sélection d'action et apprentissage par renforcement.

2.1. *Perception : de l'état à l'observation*

Nous considérons un système constitué d'un agent et de l'environnement dans lequel il évolue, l'environnement étant composé d'objets de types définis et reconnaissables par l'agent. A chaque instant, l'état du système est donné par l'état de l'agent et des objets du système. L'agent ne perçoit que partiellement le système à travers une liste de percepts liés à chaque objet. Ainsi, un agent perçoit la scène courante (la partie du système qui lui est accessible) comme une **observation** qui est un ensemble d'objets **typés** définis par leurs **percepts**.

En notant $\mathfrak{T}$ l'ensemble des types d'objets présents dans le système, nous allons spécifier quelques concepts utilisés par la suite :

– objet perçu obj : défini par son type et la perception courante que l'agent en a : $obj = [t, per]$ où $t \in \mathfrak{T}$. Nous noterons $t(obj)$ et $\mathcal{O}(obj)$ le type et la perception courante de l'objet ;

– observation o : ensemble d'objets (typés) actuellement perçus par l'agent : $o = \{obj_i\}$;

– configuration c : une configuration est un sous-ensemble ordonné d'objets de l'observation. L'ordre est important, car deux objets de même type peuvent avoir des rôles différents ;

– type de configuration C^T : un type de configuration $C^T = \langle t_1, \ldots t_n \rangle$ est une liste de types d'objets (non nécessairement uniques). Cela nous permet de définir un opérateur C^T qui extrait d'une observation donnée o toutes les configurations *compatibles* avec le type de configuration : $\mathcal{C}^T(o) = \{c \subseteq o$ telle qu'il existe une application bijective $m : C^T \rightarrow c\}$;

– observation d'une configuration : liste des perceptions associées avec les objets d'une configuration c : $o(c) = (\mathcal{O}(obj), obj \in c)$.

2.2. *Motivation liée à un comportement*

L'architecture dont nous souhaitons doter notre agent suppose que celui-ci connaisse un certain nombre de comportements dits "de base" répondant chacun à une motivation. En ce sens, il faut en premier lieu définir ce qui constitue une motivation, puisqu'un comportement sera simplement le résultat d'un apprentissage guidé par une motivation.

Un premier aspect est naturellement lié aux signaux de récompense. Pour ne pas être trop restrictifs, une première caractéristique d'une motivation sera un sous-ensemble (non vide) parmi l'ensemble des signaux de récompense élémentaires disponibles. Dans le cas du monde des tuiles, cela revient à choisir un ou deux signaux de récompense parmi les deux disponibles : un pour pousser des tuiles dans des trous (noté $+$) et un pour éviter des trous (noté $-$).

Les comportements que nous pouvons concevoir étant lié à un nombre fixe de percepts *i.e.* d'objets dans notre exemple), il nous faut aussi préciser les types et nombres des objets à prendre en compte (ces types étant instanciés dans un monde donné). Si l'agent ne vise qu'à pousser des tuiles dans des trous, il peut être intéressant pour lui de considérer un trou et deux tuiles simultanément, sans quoi l'une des deux tuiles peut devenir un obstacle non-perçu.

Dans le présent article, une motivation m est ainsi décrite par :

1) $r = \{re_1, \ldots, re_k\}$ un sous-ensemble de signaux de récompense élémentaires et

2) $C^T = \langle t_1, \ldots t_n \rangle$ un type de configuration,

ce qui est représenté sur la figure 4. Cette motivation va guider des approches d'apprentissage par renforcement.

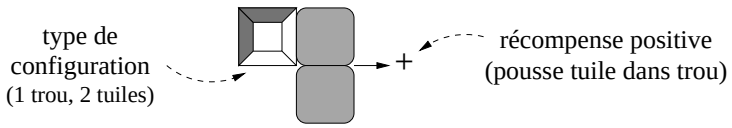


Figure 4. *Une représentation schématique d'une motivation pour 1 trou, 2 tuiles et une récompense positive ($+$) quand une tuile est poussée dans un trou.*
La position relative des objets n'a aucune sorte d'importance. Seuls leur type et leur nombre comptent

2.3. *Sélection d'action*

La sélection d'action pose le problème très général de choisir l'action à accomplir face à différents buts parallèles qui peuvent être hétérogènes et même conflictuels, et répond donc en ce sens à la problématique que nous nous posons. Mais si les

éthologues ont étudié ce problème en profondeur, leurs modèles s'avèrent difficiles à implémenter, comme l'explique Toby Tyrrell (Tyrrell, 1993).

2.3.1. *Problème posé*

Un grand nombre de modèles de sélection d'action mis en œuvre évaluent la qualité d'une action par rapport aux divers buts de haut niveau, se basant sur une somme pondérée de diverses quantités. Le processus obtenu peut être vu comme la consultation de divers experts pour prendre une décision faisant un compromis entre leurs avis (voir figure 5). Dans notre cas, il va falloir identifier dans l'environnement courant les motivations présentes (comme définies dans la section 2.2) pour associer à chacune un comportement de base qui jouera le rôle d'expert.

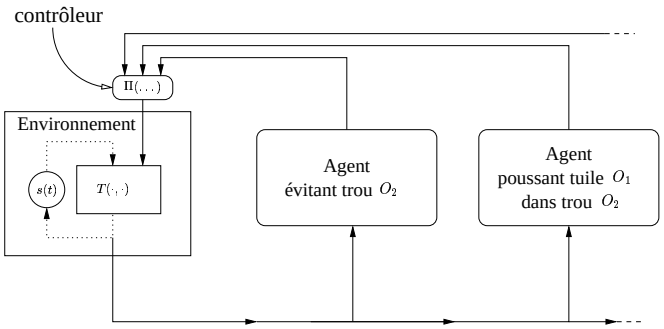


Figure 5. *Une schéma d'architecture de sélection d'action pour le monde des tuiles. Le "contrôleur" doit faire un compromis entre les propositions des différents experts*

Pour éviter la tâche laborieuse de régler manuellement un grand nombre de paramètres associés à ce processus de pesée et de sélection d'actions, diverses approches d'*apprentissage par renforcement* (A/R) ont été étudiées (voir (Lin., 1992, Humphrys, 1996) par exemple). Le présent article se situe dans cette lignée, proposant une approche innovante pour atteindre une plus grande automatisation de la conception de l'agent. Comme nous l'avons déjà dit en introduction, des méthodes existent déjà pour apprendre les comportements de base (l'A/R classique peut suffire) et pour adapter leur combinaison (l'A/R permet aussi d'optimiser les paramètres du contrôleur). Nous proposons en plus de *trouver quels comportements de base sont utiles*.

2.3.2. *Notre cadre de sélection d'action*

Dans sa thèse (Tyrrell, 1993), Toby Tyrrell a fait une analyse des différentes caractéristiques principales des architectures de sélection d'action. Nous en tirons deux choix importants pour notre propre architecture :

– le processus de sélection d'action est de type "flux-libre" : l'action choisie peut être différente des actions recommandées par les comportements de base. L'agent est capable de combiner les comportements en une *nouvelle* décision (ex : une action peut

être classée deuxième par tous les experts, et faire ainsi l'unanimité).
Ceci s'oppose aux processus "winner-takes-all" où *un* expert (donc un comportement) est élu et prend seul la décision ;

– de plus, on ne peut raisonnablement obtenir un contrôleur déterministe, prenant toujours la même décision face à une observation donnée. On risquerait alors trop facilement de rester bloqué dans une situation à laquelle la réponse associée est mauvaise. Il est donc préférable de travailler avec des comportements stochastiques.
Ce problème est similaire à celui des problèmes de décision markoviens *partiellement observables* dans lesquels la propriété de Markov est perdue (voir section 2.4).

2.4. *Notre cadre d'apprentissage par renforcement*

2.4.1. *Problèmes de décision markoviens partiellement observables*

Le problème que nous voulons résoudre dans le cadre de l'apprentissage par renforcement (Kaelbling *et al.,* 1996) peut se formaliser comme un problème de décision markovien partiellement observable (POMDP) (Littman *et al.,* 1995, Cassandra, 1998). Cette généralisation des problèmes de décision markoviens (MDP) (Sutton *et al.,* 1998, Bertsekas *et al.,* 1996, Puterman, 1994) prend explicitement en compte le fait que le système n'est que partiellement perçu par l'agent qui cherche à le contrôler.

Comme schématisé en figure 6, un POMDP est défini par un uplet $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, \mathcal{O}, r \rangle$. Ici, $\mathcal{S}$ est un ensemble fini d'**états** du système, Ω un ensemble fini d'**observations** possibles et $\mathcal{A}$ un ensemble fini d'**actions**. La dynamique du système est décrite par une **fonction de transition** $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ où $\mathcal{T}(s, a, s') = Pr(S_{t+1} = s' | A_t = a, S_t = s)$, et le lien entre état et observation se traduit par la **fonction d'observation** $\mathcal{O}(s, o) = Pr(O_t = o | S_t = s)$. Enfin, r est une application de $\mathcal{S} \times \mathcal{A}$ dans $\mathbb{R}$ qui définit la **récompense** obtenue par le système après chaque transition d'état.

L'agent, qui n'a accès qu'aux observations, doit trouver une **politique** d'action (par exemple sous la forme $\pi : \Omega \rightarrow \Pi(\mathcal{A})$) qui optimise une mesure de performance basée sur la récompense appelée valeur et notée V. Typiquement, la valeur peut être la somme des récompenses sur un horizon fini, la somme pondérée de ces récompenses sur un horizon infini ($\sum_{t=0}^{\infty} \gamma^t r_t$ où $\gamma \in [0, 1)$), ou la récompense moyenne obtenue pendant une transition (*i.e.* pendant un pas de temps).

Dans le cas où l'observation du système est totale (une observation correspond sans équivoque à un état), il a été prouvé qu'il existe au moins une politique optimale déterministe et l'agent peut alors la trouver (Puterman, 1994). Si, comme dans notre cas, l'agent ne connaît pas la dynamique du système, il peut directement apprendre une politique optimale (en utilisant le Q-learning (Watkins, 1989) ou SARSA (Sutton *et al.,* 1998) par exemple) à partir de ses expériences.

Toutefois, dans notre situation, un agent ne perçoit que des observations partielles et, de *son* point de vue, le problème de décision n'est pas markovien. Il peut donc ne

pas y avoir de politique déterministe parmi les solutions optimales. Les algorithmes de programmation dynamique mentionnés précédemment ne sont plus adaptés à ce nouveau cadre, mais peuvent être remplacés par des algorithmes de recherche directe de politiques tels que la descente de gradient en ligne que nous avons utilisée (due à (Baxter *et al.,* 2001a, Baxter *et al.,* 2001b)).

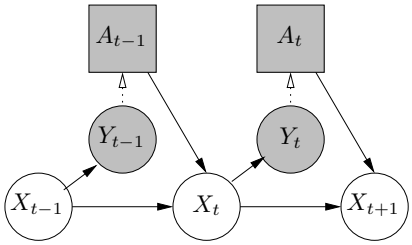


Figure 6. *Modèle graphique d'un POMDP. Les flèches pointillées indiquent que le choix d'une action dépend de l'état courant. Ici, les états sont cachés (représentés avec un fond blanc), contrairement aux observations et actions (fond gris). Les récompenses ne sont pas représentées, afin de garder une figure lisible.*
Note : les décisions prises ne sont ici dépendantes que de l'observation courante

L'apprentissage est d'autant plus difficile dans ces nouvelles conditions, du fait de possibles optima locaux dans lesquels l'agent peut tomber définitivement. Les lecteurs intéressés peuvent se référer à (Littman *et al.,* 1995) et (Singh *et al.,* 1994) sur le sujet de l'utilisation dans un cadre *partiellement* observable d'algorithmes dédiés à un cadre *totalement* observable, et à (Jaakkola *et al.,* 1994) pour une alternative à la descente de gradient.

2.4.2. *La descente de gradient de Baxter et Bartlett*

Sans entrer dans les détails de cette descente de gradient en-ligne, nous décrivons ici succinctement son principe et les conditions dans lesquelles elle a été utilisée.

Comme proposé dans (Baxter *et al.,* 2001a, Baxter *et al.,* 2001b), une politique π dépend d'un ensemble de paramètres Θ. Ceci conduit à une valeur espérée $V(\pi_\Theta) = V(\Theta)$. La descente de gradient permet d'obtenir une politique *localement* optimale en cherchant un Θ^* qui annule le gradient $\nabla V(\Theta)$.

L'ensemble de paramètres que nous choisissons est $\Theta = \{\theta(o,a),\ (o,a) \in \Omega \times \mathcal{A}\}$, où tout $\theta(o,a)$ est un paramètre à valeurs réelles. La politique est définie par les probabilités d'effectuer l'action a quand o est observé par :

$$\pi_\Theta(o,a) = \frac{e^{\theta(o,a)}}{\sum_{b \in \mathcal{A}} e^{\theta(o,b)}}$$

A l'aide de cet algorithme, la donnée d'une motivation (définie en section 2.2) permet d'apprendre la politique qui servira au comportement de base correspondant dans notre mécanisme de sélection d'action (introduit en section 2.3).

3. Combiner des comportements de base

Nous présentons ici brièvement le mécanisme de sélection d'action (apprentissage et combinaison de comportements) utilisé dans cet article. Différentes variantes ont été envisagées, dont une analyse est faite dans (Buffet, 2003) (ainsi qu'une présentation plus complète).

3.1. *Comportements de base*

Pour une motivation donnée, tout algorithme d'A/R adapté aux observations partielles peut être utilisé. Il n'est requis que d'avoir une motivation, c'est à dire :

– de définir une fonction de récompense "guide" (comme la somme de fonctions de récompenses élémentaires sélectionnées), et

– de construire la perception basée sur les observations des objets (objets correspondants au type de configuration).

Le premier résultat obtenu est une politique stochastique (à l'aide de l'algorithme vu en section 2.4.2). Néanmoins, il faudra mémoriser d'autres informations pour faire usage de ce "comportement". De la motivation de départ, on va retenir le type de configuration (qui servira à trouver les objets adéquats dans l'environnement), mais pas la fonction de récompense : on se contentera de la fonction qualité Q (qui aidera à peser l'importance du comportement en fonction de l'espérance de gain).

Un **comportement** b est ainsi défini par un uplet $\langle \mathcal{C}_b^T, P_b, Q_b \rangle$, où :

1) $\mathcal{C}_b^T$ est le type de configuration : l'uplet des types d'objets impliqués dans le comportement ;

2) $P_b(a|o(c))$ est la politique de décision stochastique. A toute observation d'une configuration, elle associe une distribution de probabilité sur les actions ;

3) $Q_b(o(c), a)$ est la Q-table de cette politique, donnant l'espérance de récompense actualisée d'une paire (observation, action).

Un tel comportement répond à une motivation donnée. Mais notre objectif est de trouver un ensemble de comportements qui puissent être combinés ensemble pour prendre une décision. Ces comportements sélectionnés sont nommés **comportements de base**, notés ***bb*** (acronyme de "*basic behavior*") et constituent un ensemble noté $\mathcal{B}$. Avant de voir comment choisir cet ensemble, la prochaine section présente le processus de sélection d'action utilisé.

3.2. *Principe de notre architecture de sélection d'action*

La prise de décision consiste en deux étapes. D'abord, l'agent doit identifier les ensembles d'objets déclenchant des comportements de base. Ensuite, les décisions stochastiques correspondantes sont combinées en une seule faisant un compromis.

La figure 3 illustre ce processus avec seulement deux bb intuitifs :

1) b_p qui considère une tuile et un trou, et la récompense pour pousser une tuile dans un trou, et

2) b_e qui considère un trou, et la récompense pour tomber dans un trou.

On a alors deux raisons pour utiliser b_p : les configurations (uplets d'objets) $cfg_1 = \langle O_1, O_2 \rangle$ et $cfg_2 = \langle O_3, O_2 \rangle$, et une raison pour utiliser b_e : la configuration $cfg_3 = \langle O_2 \rangle$. A chaque paire (bb, cfg) sont associées une distribution de probabilité sur les actions et les Q-valeurs associées, tout cela étant employé pour calculer une distribution de probabilité finale à appliquer, comme montré sur la figure 7.

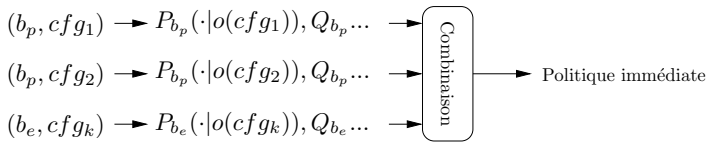


Figure 7. *Comment obtenir une distribution de probabilité sur les actions avec les trois paires* (bb, cfg) *identifiées*

Les deux sections suivantes décrivent plus en détail les principales opérations que sont la décomposition de la scène et la combinaison de comportement elle-même.

3.2.1. *Décomposition de la scène*

L'objectif de cette décomposition est de trouver quels sont les comportements de base applicables dans une situation donnée.

L'environnement de l'agent est constitué d'objets typés (tels que les trous et tuiles de la figure 3). L'observation de l'agent est un ensemble d'objets perçus dans l'environnement. Chaque comportement de base n'est concerné que par certains types d'objets. Ainsi, ils sont "instanciés" en les associant avec des configurations (cfg), *i.e.* des sous-ensembles d'objets de types appropriés. Cela amène à une décomposition de la scène en un ensemble $\mathcal{BC}(o)$ de paires $(comportement\ de\ base, configuration)$. Revenant à la figure 3, nous avons ici trois telles paires : $(b_e, \langle O_2 \rangle)$, $(b_p, \langle O_1, O_2 \rangle)$ et $(b_p, \langle O_3, O_2 \rangle)$.

Nous avons vu comment chaque scène est décomposée en paires $(bb,\ cfg)$. On a une politique de décision stochastique ($P_{bb}()$) pour chaque paire. Et, comme chaque

observation est associée avec une observation $o(cfg)$, chaque paire fournit une distribution de probabilité sur les actions $P_{bb}(a|o(cfg))$.

3.2.2. *Combinaison pondérée de comportements de base*

Chaque comportement de base bb de $\mathcal{BC}(o)$ identifié lors de la décomposition de la scène est associé avec une politique d'action $P_{bb}()$ et une qualité $Q_{bb}()$. Avec ces données, l'étape de combinaison consiste à calculer une nouvelle distribution de probabilité sur les actions, déterminant ainsi la politique réelle immédiate de l'agent.

La combinaison peut prendre des formes diverses. Celle que nous avons employée ici dépend d'un ensemble $\Theta = (\theta_1, \cdots, \theta_n)$ de paramètres et est exprimée par la formule suivante (où $\mathcal{C}(b, o)$ est l'ensemble des configurations observées liées à b) :

$$Pr(a|o) = \frac{1}{K}.\frac{1}{k_{(o,a)}} \sum_{b \in \mathcal{B}} \underbrace{e^{\theta_b}}_{\text{à apprendre}} \underbrace{[\sum_{c \in \mathcal{C}(b,o)} |Q_b(o,c,a)|.P_b(o,c,a)\,]}_{\text{déjà connu}}$$

$$(\text{ avec } k_{(o,a)} = \sum_{(b,c) \in \mathcal{BC}(o)} w_b(o,c,a) = \sum_{(b,c) \in \mathcal{BC}(o)} e^{\theta_b}.|Q_b(o,c,a)|\,)$$

Apprentissage des poids

La politique combinée globale ne dépend que de l'ensemble Θ de paramètres, lesquels sont peu nombreux puisqu'il n'y a qu'**un paramètre pour chaque comportement de base**.

Comme le réglage de ces poids est un problème d'optimisation (maximiser l'espérance du gain moyen), ces paramètres peuvent être appris par renforcement à l'aide d'un recuit simulé. L'algorithme a juste à faire une bonne estimation de l'espérance du gain moyen pour chaque jeu de paramètres.

On notera que ce processus de sélection d'action est "extensible" puisqu'on peut utiliser plusieurs instances d'un même bb sans apprendre de nouveaux paramètres.

3.3. *Problèmes ouverts*

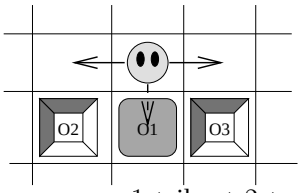
L'algorithme que nous venons de brièvement présenter a été testé sur le monde des tuiles dans (Buffet *et al.*, 2002, Buffet, 2003) en employant deux bb "intuitifs" (`[pousse tuile dans trou]` et `[évite trou]`). Malgré de "bons résultats", l'optimalité n'était pas atteinte et plusieurs limitations étaient citées. Nous insistons sur les suivantes :

– **Optima locaux**. L'apprentissage de paramètres est difficile du fait des nombreux optima locaux vers lesquels le recuit simulé peut converger. De plus, cet apprentissage

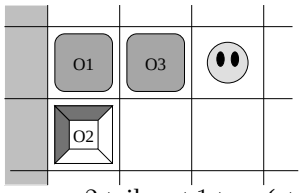
reste long malgré le faible nombre de paramètres à apprendre : il faut prendre le temps de bien évaluer le comportement obtenu avec un jeu de paramètres donné.

– **Méthode de combinaison**. Combiner des avis de divers experts est un problème très ouvert (voir (Genest *et al.*, 1986) par exemple). Les différentes méthodes évaluées dans (Buffet, 2003) reflètent en partie leur diversité et l'attention qu'il faut porter à ce choix. Nous considérons ici qu'une méthode satisfaisante a été choisie.

– **Choix des comportements de base**. L'ensemble des comportements de base disponibles a une grande influence sur les performances de l'algorithme de combinaison. Des bb très généraux (souvent intuitifs) sont requis pour avoir un agent adaptable, mais d'autres, plus spécialisés, permettraient de répondre à des cas difficiles précis (voir les deux blocages de la figure 8). Choisir les comportements de base est précisément le sujet du présent article.



a) Blocage avec 1 tuile et 2 trous : Aucun des comportements de base ne suggère d'aller vers le sud



b) Blocage avec 2 tuiles et 1 trou (et un mur est) : Les deux bb "pousse" suggèrent d'aller à l'ouest, alors que le bon choix serait le nord

Figure 8. *Dans les deux situations a) et b), aucun des comportements de base "intuitifs" employés ne suggère une décision appropriée, d'où un long blocage de l'agent, et donc une grande perte d'efficacité*

4. Quels comportements de base utiliser ?

4.1. *Comment détecter un nouveau comportement de base*

En quelques mots, la construction d'un nouveau comportement de base à partir d'un ensemble existant de bb repose sur trois étapes :

1) apprendre le meilleur comportement combiné pour une motivation donnée,

2) essayer d'améliorer ce comportement et

3) décider s'il sera alors utilisé comme un nouveau comportement de base.

4.1.1. *Apprendre à combiner des comportements de base*

Etant donné un ensemble de comportements de base $\mathcal{B} = \{bb_1, \cdots, bb_n\}$, on sait donc adapter efficacement la formule qui les combine en optimisant les paramètres de réglage θ_b. Le comportement résultant de ce processus d'apprentissage est un *compor-*

tement combiné[1] : $cb = \mathcal{F}(\mathcal{B}, \Theta)$ où $\Theta = (\theta_1, \cdots, \theta_n)$ est un vecteur de paramètres appris de la taille de $\mathcal{B}$.

On constate malheureusement que les choix intuitifs de comportements de base ne permettent pas de répondre à certaines situations de blocage particulières (voir figure 8), lesquelles sont peu nombreuses (faible pourcentage des états possibles), mais font perdre un temps précieux (fort pourcentage du temps de l'agent).

4.1.2. *Amélioration*

Sachant que seules peu de situations posent réellement problème, on en déduit que le *comportement combiné* cb obtenu pour la motivation $m = (r, C^T)$ choisie est très proche d'un comportement très efficace, et n'aurait besoin que de quelques corrections localisées. De là est venue l'idée de partir d'un tel comportement combiné pour apprendre un *nouveau comportement de base* nb, l'apprentissage ayant principalement à corriger les quelques décisions erronées qui font perdre le plus de temps[2].

Pour ce faire, une descente de gradient "par renforcement" en ligne (Baxter *et al.*, 2001b) permet d'améliorer la politique du comportement combiné cb dont nous disposons, même dans une cadre d'A/R non markovien tel que le nôtre. De la nouvelle politique ainsi obtenue, nous pouvons dériver un nouveau comportement (noté nb) avec la motivation et la qualité associées. Ce procédé est résumé dans l'algorithme 1 et illustré par la figure 9.

Algorithme 1 Améliorer un comportement combiné en un nouveau comportement

Entrées: $\mathcal{B}$ un ensemble de comportements de base.
Une motivation $m = (r, C^T)$ (signaux de récompense + type de configuration).

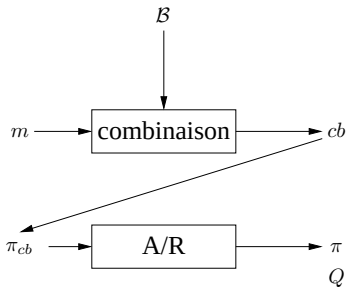
1: Adapter la combinaison des bb courants de $\mathcal{B}$ (pour la motivation m).
Résultat : π_{cb} politique associée à m.
2: Apprendre par renforcement une nouvelle politique pour m, en partant de π_{cb}.
Résultat : π politique améliorée associée à m.
3: En déduire la table de Q-valeurs associée à cette nouvelle politique.
Résultat : Q.

Sorties: Nouveau comportement nb associé à m
(de politique π et de table de Q-valeurs associée Q).

Les expérimentations effectuées (Buffet *et al.*, 2002) montrent qu'il est largement préférable de suivre cet algorithme plutôt que de faire un apprentissage partant de zéro. On peut ainsi dépasser des optima locaux (se contenter d'apprendre à éviter des trous dans notre cas), et l'apprentissage est largement accéléré.

On peut en revanche se demander si l'on n'en apprend pas trop si seules quelques situations sont problématiques. Malheureusement, il n'est pas évident de détecter ces

1. Attention : un comportement combiné n'est pas un comportement comme défi ni en section 3.1, ce n'est que le résultat d'une combinaison.
2. cb et nb sont des acronymes de l'anglais "combined behavior" et "new behavior".



Etant donné un ensemble de comportements de base $\mathcal{B}$ et une motivation m, on cherche d'abord un "comportement combiné" cb pour n'en garder que la politique π_{cb}. Puis on améliore cette politique par A/R pour obtenir un nouveau comportement de politique π.

Figure 9. *Schéma de principe de l'amélioration d'un comportement combiné*

situations, d'autant que l'on peut avoir des boucles comportementales dont l'agent ne sait sortir. Une solution envisageable serait de comparer π_{cb} avec π pour voir ce qui a été modifié par l'apprentissage par renforcement. Mais un autre aspect à considérer est le calcul des Q-valeurs : peut-on se contenter de retenir les Q-valeurs associées aux décisions nouvelles ? Nous laissons cette question en suspend, nous contentant de ce nouveau comportement complet nb.

4.1.3. *Nouveau comportement de base ? Un critère*

Supposant qu'il n'est pas évident que le nouveau comportement nb soit bien meilleur que le comportement combiné cb, la question qui se pose maintenant est de savoir si l'utilisation de nb comme comportement de base dans le mécanisme de sélection d'action améliorerait le comportement global de l'agent.

Nous proposons ici de baser cette décision sur une comparaison entre la qualité du comportement combiné et celle du nouveau comportement. L'hypothèse suivie est que, si un comportement combiné est nettement amélioré par apprentissage, le nouveau comportement obtenu apporte de nouvelles connaissances utiles.

Si on appelle (V_{cb}) la qualité du comportement combiné et (V_{nb}) la qualité du nouveau comportement, le nouveau comportement est choisi comme nouveau comportement de base (et ajouté à l'ensemble $\mathcal{B}$) si :

$(|V_{cb} - V_{nb}| > \sigma_s) \wedge$
(Permet d'éviter des erreurs dues à des estimations proches.)

$\{ [(V_{cb} < 0) \wedge (V_{nb} < \sigma_- * V_{cb})]$
(Teste s'il y a une amélioration, même faible, d'un résultat.)

$\vee [(V_{cb} \geq 0) \wedge (V_{nb} > \sigma_+ * V_{cb})] \}$
(Teste s'il y a une amélioration majeure d'un résultat positif.)

NOTE. — Le calcul de V est délicat car il consiste en une estimation par méthode de Monte-Carlo (MacKay, 2003, chapitre 29). On laisse l'agent évoluer au sein de son environnement en mesurant au fur et à mesure la récompense moyenne qu'il reçoit. Comme l'agent peut se retrouver bloqué assez longtemps dans certaines situations,

une bonne estimation demande un temps relativement long pour que de tels blocages soient pris en compte à leur juste valeur.

Ce critère dépend de trois paramètres qui ont une grande influence sur l'algorithme général, comme détaillé en section 4.2. Dans nos expérimentations, ils ont été réglés manuellement. Un travail important serait de tester leur validité sur d'autres tâches, de regarder s'ils doivent être beaucoup modifiés d'une tâche à l'autre.

4.2. *Recherche des comportements de base utiles*

On cherche ici à automatiser le choix des comportements de base. Comme expliqué en section 3.3, c'est une question majeure puisque la sélection intuitive de comportements de base "minimalistes" n'est pas suffisante. De plus, cette étape est requise pour avoir un agent réellement autonome, demandant un minimum de travail de conception.

Mais avant de présenter l'algorithme proposé, notons tout d'abord que l'on peut doter l'ensemble des types de configurations d'une relation d'ordre partielle : un type de configuration C_a^T ayant au moins les mêmes types d'objets et en nombre au moins égal qu'un autre type de configuration C_b^T est "plus grand" que ce second type. On peut de même doter d'une relation d'ordre partielle l'ensemble des sous-ensembles de fonctions de récompense élémentaires. Et, de là, on peut déduire une relation d'ordre sur les motivations : $m_a > m_b$ si et seulement si $C_a^T > C_b^T$ et $r_a > r_b$.

4.2.1. *Principe*

Pour rechercher les comportements de base jugés utiles, nous proposons de construire incrémentalement un ensemble de comportements de base. Dans ce but, partant d'un ensemble vide, on y ajoute progressivement des comportements de plus en plus complexes, en utilisant des bb déjà sélectionnés pour en concevoir de nouveaux. Ceci conduit à construire un ensemble de bb de complexité croissante. Or on vient de voir qu'on a une relation d'ordre partielle sur les motivations. Cette relation d'ordre implique que la "complexité" d'un comportement vient de deux sources (liées à la motivation associée à ce comportement) : la taille du type de configuration, et les fonctions de récompense élémentaires choisies.

Pour résumer les principes adoptés, un nouveau type de configuration étant choisi, les comportements correspondant aux possibles combinaisons de récompense sont appris et testés pour déterminer leur intérêt. La figure 10 illustre la progression selon les types de configuration : la complexité croît quand on descend dans l'arbre. Utilisant le processus décrit en section 4.1.3 pour sélectionner un nouveau comportement de base, la section suivante explique l'algorithme de croissance de l'arbre.

4.2.2. *Croissance de l'arbre*

Ne prenant en compte que les types de configurations, la relation d'ordre définie permet de les organiser sous la forme d'un treillis dans lequel les descendants directs

d'un nœud donné ont chacun un type d'objet différent en plus. Mais faire une recherche de comportements de base utiles dans ce treillis pourrait s'avérer très onéreux. On va donc orienter cette recherche en créant progressivement l'arbre[3] dans lequel est menée l'exploration.

On construit donc un arbre d'*exploration* de comportements de complexité croissante selon le nombre d'objets. Chaque nœud de l'arbre est lié à un type de configuration et est en fait un court sous-arbre des combinaisons de récompense possibles. A l'aide d'un parcours en largeur d'abord, on calcule le meilleur comportement pour chaque nœud, sur la base d'une combinaison des bb plus anciens (comme expliqué en section 4.2.1). Chaque nouveau comportement nb est comparé à la combinaison correspondante cb (critère présenté en section 4.1.3) pour déterminer s'il doit être ajouté à l'ensemble des comportements de base, l'arbre étant alors développé à ce nœud par de nouveaux comportements à explorer.

Revenant à l'exemple de la figure 10, l'arbre suit des types de configurations de plus en plus complexes, et à chacun d'eux correspond un sous-arbre des combinaisons de récompenses possibles ("$-$" (respectivement "$+$") pour la récompense négative pour l'évitement de trou (resp. la récompense positive pour pousser une tuile dans un trou), et "$+-$" qui combine les deux).

4.2.3. *Algorithme*

Avant de le tester, voyons une définition formelle de notre méthode à travers l'algorithme 2. Parmi les données requises par cet algorithme, on a :

– p_{max} : Comme une croissance sans fin de l'arbre est possible, ce paramètre limite la profondeur de l'arbre ;

– p : Les fils immédiats d'un nouveau bb ne sont pas les seuls dignes d'intérêt pour des explorations futures. Ainsi, ce second paramètre définit la profondeur de développement d'un nœud (voir la *profondeur de frange* de (McCallum, 1995)) ;

– $Critere(\cdot,\cdot)$: critère d'évaluation des nouveaux comportements (section 4.1.3).

Les expérimentations qui suivent incluent (section 5.2.1) une illustration du comportement de cet algorithme sur le problème du monde des tuiles.

5. Expérimentations

5.1. *Application au monde des tuiles*

5.1.1. *Description du problème*

Le monde des tuiles est un domaine quadrillé dans lequel une case peut contenir un trou, une tuile ou un agent. L'agent (un seul est considéré) doit pousser des tuiles dans des trous aussi souvent que possible, et évite de passer lui-même dans ces trous. Pour

3. On parlera ici abusivement d'arbre même s'il s'agit en fait d'un graphe orienté acyclique.

Algorithme 2 Un arbre croissant de comportements de base

Entrées: $Critere(\cdot,\cdot)$: pour évaluer les nouveaux comportements.
p_{max} : profondeur maximale de l'arbre.
p : profondeur des sous-arbres développés.
1: T arbre vide. $\mathcal{B}$ ensemble vide.
2: Ajouter à T tous les comportements ayant de 0 à p types d'objets.
3: **pour tout** Comportement b encore à visiter (jusqu'à la profondeur $p_{\max}$) **faire**
4: Adapter la combinaison des bb courants de $\mathcal{B}$:
$V_{cb} \leftarrow$ efficacité de ce comportement combiné cb.
5: Apprendre b par une recherche directe de politique (initialisée par cb) :
$V_{nb} \leftarrow$ efficacité de ce nouveau comportement nb.
6: **si** $Critere(V_{cb}, V_{nb}) = vrai$ **alors**
7: Ajouter à T les fils de b ayant jusqu'à p types d'objets en plus.
8: Marquer b comme `[basique]`. L'ajouter à $\mathcal{B}$.
9: **fin si**
10: Marquer b comme `[visité]`.
11: **fin pour**
Sorties: Un ensemble $\mathcal{B}$ de comportements de base retenus.

détailler la simulation, l'agent peut aller librement dans un trou (et en sortir), mais recevra alors une récompense négative. De plus, quand une tuile est poussée dans un trou, tuile et trou disparaissent pour réapparaître autre part sur la grille. Enfin, pour éviter quelques situations bloquantes, trous et tuiles ne peuvent être sur des cases du bord de la grille.

Comme les exemples précédents le montrent (figure 3), une simple décomposition du problème en comportements de base peut être faite intuitivement : 1-`[éviter trou]` $(b_e, \langle trou \rangle)$ et 2- `[pousser tuile dans trou]` $(b_p, \langle tuile,\ trou \rangle)$. Ils seront utilisées comme référence pour notre génération d'un ensemble de bb (noté $\mathcal{B}_{ref}$).

5.1.2. *Perceptions, actions et récompenses de l'agent*

L'agent voit toujours tous les objets de l'environnement. Le principe de localité n'en est pas moins respecté, du fait des perceptions imprécises (agent myope). Pour tout objet O de la scène, la *perception* de O par l'agent donne :

– **direction(O)** : donne la direction de l'objet `(N-NE-E-SE-S-SO-O-NO)`, et

– **proche(O)** : dit si l'objet est sur le carré de 9 cases centré sur l'agent `(vrai/faux)`.

Les seules *actions* possibles pour un agent sont de se déplacer d'une case vers le nord, le sud, l'est ou l'ouest (il ne peut demander à rester sur une case). Et pour finir, la *récompense* donnée est $+1$ quand une tuile tombe dans un trou, -3 quand l'agent passe dans un trou, et 0 dans tout autre cas.

Revenant à la figure 3, les perceptions des 3 objets sont ici :
O_1 : $proche(O_1) = faux$, $direction(O_1) = O$
O_2 : $proche(O_2) = faux$, $direction(O_2) = O$
O_3 : $proche(O_3) = vrai$, $direction(O_3) = S$

5.1.3. *Méthodologie*

Tout d'abord, précisons que les paramètres apparaissant dans la description de l'algorithme 2 de croissance d'arbre (y compris le critère de la section 4.1.3) ont été réglés manuellement, et définis comme suit :

$$\sigma_s = 500 \qquad \sigma_- = 0,9 \qquad \sigma_+ = 2 \qquad p_{max} = 5 \qquad p = 2$$

Pour analyser le comportement de l'algorithme proposé, les expérimentations conduites prennent deux directions :

1) appliquer l'algorithme de croissance d'arbre pour produire un ensemble de comportements de base $\mathcal{B}_{arbre}$ (dans une grille 6×6) et

2) tester son efficacité dans diverses situations plus complexes (avec plus d'objets, et dans une grille 8×8 pouvant les accueillir).

Les tests conduits dans cette seconde phase consistent à employer la combinaison des bb résultants avec différents nombres de tuiles et de trous (jusqu'à cinq de chaque). La comparaison est alors faite avec les deux bb "intuitifs" de référence (voir section 5.1.1). L'efficacité d'une combinaison est mesurée à travers le gain total reçu pendant $100\,000$ pas de simulation.

5.2. *Analyse*

5.2.1. *Croissance de l'arbre*

La figure 10 montre l'arbre des comportements évalués qui a été développé par l'algorithme de croissance d'arbre. Les signes mis entre parenthèses indiquent que le comportement lié à ce type de récompense et au type de configuration correspondant a été retenu comme comportement de base.

Profitons d'abord de cette utilisation de l'algorithme sur un exemple pour décrire son fonctionnement (suivre sur la figure 10), sachant que l'arbre n'est que de hauteur 1 (un objet au plus par nœud) au départ.

– L'agent débute sans comportement de base, ce qui résulte en des déplacements aléatoires (mouvement brownien). Placé d'abord dans un environnement sans objets (racine de l'arbre), il n'apprend rien, quels que soient les signaux de récompense considérés : son mouvement brownien fait aussi bien que tout ce qu'il peut apprendre d'autre.

– On place ensuite ce même agent dans un environnement avec un objet. Commençant avec un trou, il va se rendre compte qu'il peut apprendre un comportement utile permettant d'éviter un trou mieux que ne le fait le mouvement brownien (on retient le

comportement et étend l'arbre localement, ajoutant un trou d'un côté, et une tuile de l'autre). Par contre, considérer une seule tuile n'amène rien d'intéressant.

– Les situations à deux objets contiennent toutes deux un trou. Si savoir éviter un trou (le premier *bb* retenu) permet d'en éviter deux, on peut par contre ici apprendre à pousser une tuile dans un trou. Cela donne un second comportement de base (et une nouvelle extension de l'arbre).

– L'agent continue ainsi son apprentissage, ne réussissant à résoudre que l'une des deux situations de blocage à 3 objets...

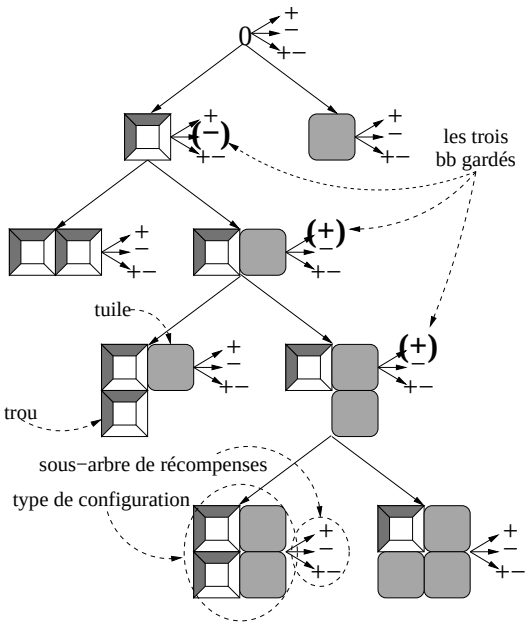


Figure 10. *L'arbre des comportements testés et (entre parenthèses) ceux qui ont été gardés. Les arcs indiquent quels sous-arbres sont développés depuis chaque nœud, sachant que plusieurs nœuds pourraient être à l'origine du développement d'un même sous-arbre*

Les trois *bb* ainsi obtenus (ensemble $\mathcal{B}_{arbre}$) correspondent aux deux *bb* intuitifs (de $\mathcal{B}_{ref}$) avec le comportement [1-trou/2-tuiles avec récompense "$+$"] "spécialisé" dans la résolution d'un blocage (montré figure 8 b)). L'arbre généré semble satisfaisant puisqu'ajouter le comportement "spécialisé" répond à une situation de blocage typique.

Pourtant, le comportement [2-trous/1-tuile, récompense "$+$"] qui résout un autre blocage (figure 8 a)) n'est pas ajouté à l'ensemble des comportements de base. En fait, l'amélioration des performances qu'il aurait amené ne satisfait pas le critère ($V_{nb} \simeq 1,5 * V_{cb} < 2 * V_{cb}$), ce qui explique qu'il soit abandonné. Si l'on utilise un critère "plus souple" ($\sigma_+ = \mathbf{1}, \mathbf{3}$), les deux *bb* spécialisés sont inclus dans $\mathcal{B}$, ainsi que quatre

autres plus complexes. Cela amène d'autres difficultés, le processus d'apprentissage de combinaison étant plus complexe et sujet à tomber dans des optima locaux, même si chacun de ces comportements répond à une situation spécifique intéressante.

5.2.2. *Efficacité de l'arbre*

Pour donner une première idée du problème que pose le monde des tuiles, les figures 11 a) et b) montrent l'efficacité obtenue par un agent se promenant aléatoirement (mouvement brownien) d'une part, et un agent utilisant dans chaque cas un apprentissage par renforcement simple (descente de gradient) d'autre part. Les axes x et y du plan horizontal indiquent le nombre de trous et de tuiles dans les situations évaluées (le nombre d'objets à gérer a contraint à étendre la grille à une taille de 8×8). Sur l'axe z est mesurée l'efficacité dans chaque situation (gain moyen sur 100 000 pas de temps). On constate qu'il s'avère très simple de beaucoup tomber dans des trous sans pousser la moindre tuile dans un trou (agent brownien), et que l'apprentissage par renforcement, à part dans un monde très simple (une tuile et un trou), ne permet que d'éviter les trous.

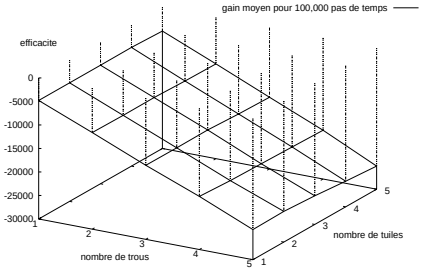
Les figures 11 c) et d) présentent l'efficacité des deux ensembles de comportements de base : $\mathcal{B}_{ref}$ et $\mathcal{B}_{arbre}$, pour pouvoir comparer le second (généré automatiquement par notre algorithme) avec le premier (conçu à la main). Pour certains points, des valeurs numériques sont présentées dans le tableau 1.

A première vue, les deux surfaces sont assez similaires, avec un pic commun dans la situation 1-tuile/1-trou (à laquelle un *bb* "pousser" est dédié). Une amélioration importante ($+30\%$) peut être observée pour 2-tuiles/1-trou, ce qui était attendu puisque $\mathcal{B}_{ref}$ et $\mathcal{B}_{arbre}$ ne diffèrent que par le comportement gérant la motivation "pousser" correspondante. Toutefois, cette amélioration persiste avec un nombre croissant de tuiles à pousser (résultats multipliés par 2) selon l'axe des tuiles (voir figure 12).

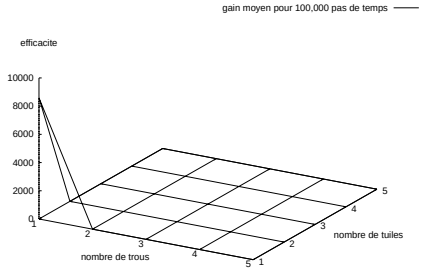
Considérant un nombre croissant de trous, les résultats de la figure 11 d) sont toujours plus proches de la surface de référence 11 c). On a vu qu'un problème lié au critère empêche les améliorations dans ces cas à deux trous et plus. De plus, les variations que l'on peut observer sont partiellement dues à l'estimation de l'efficacité.

5.2.3. *Observation d'un critère plus souple*

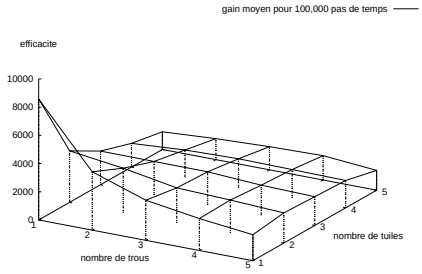
La figure 11 e) montre les résultats obtenus avec le critère plus souple que nous avons déjà mentionné ($V_{nb} > \mathbf{1}, \mathbf{3} * V_{cb}$) et une profondeur maximale de l'arbre $p_{max} = 4$. Ici, le comportement du nœud [2-trous/1-tuile avec récompense "+"] est maintenant retenu dans l'ensemble $\mathcal{B}$. Cela produit sur l'axe des trous un gain similaire à celui observé auparavant sur l'axe des tuiles. Avec ce dernier ensemble $\mathcal{B}'_{arbre}$, l'agent peut même bénéficier des deux améliorations simultanément, comme le montre le tableau 1. Ce même tableau illustre le fait qu'avoir des *bb* inutiles amène une légère perte d'efficacité (voir $\mathcal{B}_{arbre}$ dans les cas 2/1 ou 2/2, ou $\mathcal{B}'_{arbre}$ dans le cas 1/2). Elle s'explique par le fait que ces *bb* inutiles ne sont qu'un bruit ajouté dans la combinaison, perturbant ainsi la prise de décision.



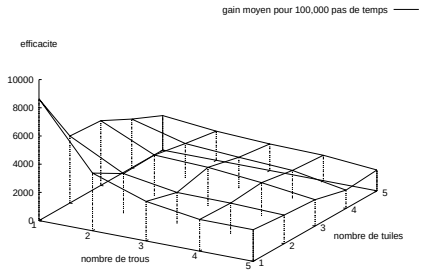
a) déplacement aléatoire (brownien)



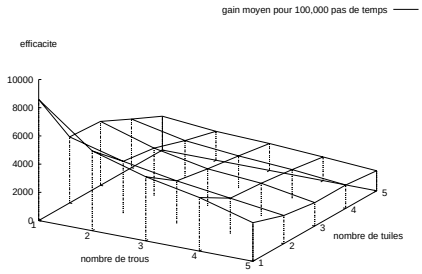
b) apprentissage par renforcement simple



c) comportements de base de $\mathcal{B}_{ref}$



d) comportements de base de $\mathcal{B}_{arbre}$



e) comportements de base de $\mathcal{B}'_{arbre}$

Notes :

– a) $\mathcal{B}_{\emptyset}$= mvt brownien.

– b) Résultat avec un apprentissage par renforcement (descente de gradient) partant de zéro dans chaque cas.

– e) Tests identiques, mais avec $\mathcal{B}'_{arbre}$ = $\mathcal{B}_{arbre}$ + le comportement de base [2-trous/1-tuile avec la récompense "+"].

Figure 11. *Efficacité de la combinaison (avec différents ensembles de comportements de base) ou de l'apprentissage par renforcement dans des situations plus complexes*

$\#trous/\#tuiles$	$\mathcal{B}_{ref}$	$\mathcal{B}_{arbre}$	$\mathcal{B}'_{arbre}$
1 / 2	3648,3	**4747,5**	4680,5
2 / 1	4137,8	4089,6	**5660,8**
2 / 2	3142,3	2807,5	**3673,4**
2 / 3	2369,6	2873,9	**3356,4**
3 / 2	2468,3	2190,3	**3007,7**

Tableau 1. *Comparaison des efficacités des trois ensembles. Avec plus de cinq objets, le gain devient plus difficile à apprécier*

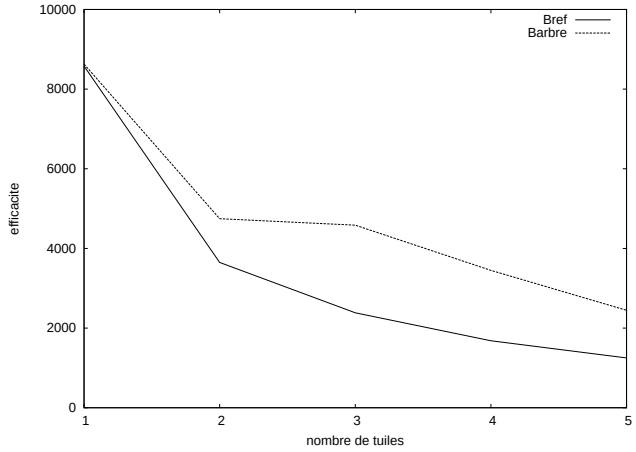
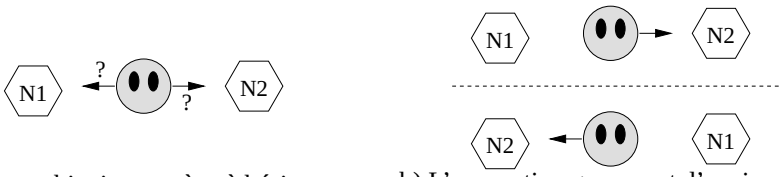


Figure 12. *Comparaison des efficacités de $\mathcal{B}_{ref}$ et $\mathcal{B}_{arbre}$ avec un trou et plusieurs tuiles*

Si la profondeur maximale de l'arbre a été limitée ici à $p_{\max} = 4$, c'est que sans cela l'algorithme trouve de trop nombreux comportements éligibles. En observant les comportements des agents, on se rend 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.

Si l'on regarde l'exemple de la figure 13 où un agent peut être attiré par deux sources de nourriture N_1 et N_2, combiner deux instances d'un comportement allant vers une source de nourriture va laisser l'agent indécis, alors qu'apprendre un comportement considérant les *deux* sources de nourriture va lui permettre "d'apprendre à fixer son attention" (toujours sur la source N_2 par exemple).



a) La combinaison amène à hésiter et osciller

b) L'apprentissage permet d'avoir un but persistant

Figure 13. *Apparition d'un phénomène de persistance de l'attention*

6. Discussion, travaux similaires

Si l'on revient aux motivations de notre travail, le but principal de la sélection d'action (être capable de faire face à de multiples tâches) a été atteint. Rappelons que les outils d'apprentissage par renforcement classiques ne s'avèrent pas capables de gérer plus d'une tuile et un trou simultanément (voir figure 11-b) ou les expérimentations dans (Buffet *et al.*, 2002)). Un agent utilisant l'A/R classique ne réussit qu'à éviter de tomber dans les trous, ce qui s'observe par la surface gisant avec un gain nul, mis à part le pic obtenu pour une tuile et un trou.

Avec notre méthode, l'agent reste capable de se comporter correctement avec jusqu'à 5 trous et 5 tuiles. C'est parce qu'elle n'essaye pas de résoudre plusieurs problèmes (répondre à plusieurs motivations) à la fois sans avoir attaqué chaque problème indépendamment dans un premier temps que notre méthode s'en sort. Evidemment, il y a un compromis à trouver entre un nombre réduit de comportements de base (apprentissage plus rapide) et des performances élevées.

Ce qui n'est pas encore automatisé

Le résultat de notre algorithme dépend toujours de paramètres et en particulier d'un critère heuristique. Les paramètres sont liés à l'exploration et à la croissance de l'arbre de comportements. Comme vu en section 5, l'influence du critère peut être décisive. Si l'on conserve ce type de critère, un processus plus intelligent serait d'adapter les paramètres σ pour partir d'un critère souple et le voir se contraindre quand le nombre de comportements de base croît. Ainsi, de nouveaux comportements ne seraient ajoutés que s'ils ont réellement un apport en termes de performances. Bien entendu, des voies d'exploration importantes sont la recherche d'un critère plus pertinent ou celle d'une façon d'automatiser le réglage des paramètres.

Pour toujours considérer des aspects pratiques, notre approche n'appartient pas complètement à l'apprentissage par renforcement *avec* ou *sans* modèle. Si l'algorithme élémentaire d'apprentissage utilisé (la descente de gradient) est dans la classe "sans modèle", l'algorithme de croissance arborescente requiert le passage par un environnement simulé (une forme de modèle), de manière à pouvoir conduire des expérimentations contrôlées. Parmi les apprentissages *avec simulation* (Kearns *et al.*, 2002), on peut citer les travaux de Péret et Garcia (Peret *et al.*, 2004), lesquels ne nécessitent pas de mémoriser un modèle complet (états, transitions...) ou une politique complète, chose qui pourrait s'avérer irréaliste.

Une simulation étant une forme de modèle du système, on se retrouve donc avec le problème d'obtenir un tel modèle dans le cas où l'on travaille sur une application réelle. Il faut alors l'intervention d'un concepteur humain ou l'utilisation d'une méthode d'apprentissage artificiel. On aborde là une autre vaste problématique d'une grande difficulté.

Travaux similaires

Notre approche a été présentée comme une alternative à l'apprentissage par renforcement relationnel dans un cadre partiellement observable. Cette idée pourrait être poussée plus loin en comparant le comportement des deux approches (l'A/R relationnel et la nôtre) sur des problèmes complètement observables. On devrait *a priori* y voir des points communs, mais aussi les limitations de notre cadre plus "souple".

Il serait intéressant de tester et comparer nos travaux sur des projets comparables. Le "house environment" conçu et utilisé dans (Humphrys, 1996) est un exemple qui s'impose, du fait que le travail de Humphrys est aussi l'un des rares à combiner sélection d'action et apprentissage par renforcement. C'est toutefois un problème à la fois :

– plus simple (que notre monde des tuiles) parce qu'il n'y a pas besoin de plusieurs instances du même comportement de base, mais en même temps,

– plus complexe à cause du facteur de branchement élevé qu'il induirait dans l'arbre d'exploration (dû à de nombreux types de récompenses (14) et d'objets (8)).

De premiers résultats montrent que les premiers comportements de base appris sont différents des bb définis manuellement par Humphrys, mais nous n'avons pas les résultats complets pour comparer les performances globales des deux approches.

En élargissant notre champ d'investigation, on peut distinguer deux principales sortes de hiérarchies dans les contrôleurs :

– des hiérarchies "parallèles" comme la nôtre ou toutes celles relevant du domaine de la sélection d'action – plusieurs motivations simultanées guidant l'agent, ce qui implique de faire un compromis – , et

– des hiérarchies "séquentielles" qui ne vont s'attaquer qu'à une tâche élémentaire à la fois, ces tâches (ex : trouver une clef) pouvant être les étapes d'une tâche de plus haut niveau (ouvrir une porte), seule cette dernière amenant une récompense.

Les hiérarchies séquentielles sont plus courantes et mieux maîtrisées dans la mesure où elles n'essayent pas de faire plusieurs choses à la fois. Dans le monde de l'apprentissage par renforcement, on peut citer de nombreux travaux (Mahadevan *et al.*, 1992, Kaelbling, 1993, Lin, 1993, Dayan *et al.*, 1993, Parr, 1998, Hauskretch *et al.*, 1998, Dietterich, 2000). Dans les deux problèmes, on retrouve la même difficulté pour établir la hiérarchie elle-même. Les algorithmes proposés par Digney (Digney, 1998) (nested Q-learning) et Hengst (Hengst, 2002) (HEXQ), par exemple, permettent de construire des hiérarchies séquentielles, ce que nous faisons pour notre part dans le cadre de hiérarchies parallèles.

En fait, dans notre travail, apprendre à la fois des comportements de base et comment les combiner a permis d'accroître l'autonomie de l'agent. Evidemment, la main de l'homme est toujours requise, en particulier pour indiquer les différents types d'objets et de récompenses liés à l'environnement. C'est aussi le cas dans (Digney, 1998), et semble inévitable du point de vue de (Urzelai *et al.*, 1998). Les travaux de ces

derniers visent précisément à trouver le juste équilibre entre conception manuelle et apprentissage artificiel pour concevoir efficacement des agents autonomes. Ils argumentent que l'humain est plus efficace pour définir des comportements de base, même s'ils sont grossiers et raffinés ultérieurement par l'agent. Notre travail, comme ceux de (Digney, 1998) voire de (Nehmzow *et al.*, 1993), semble réduire la tâche du concepteur humain : il n'a à fournir que des types de récompenses et de perceptions. Enfin, de telles approches sont à rapprocher du développement cognitif (Weng, 2002) par leur aspect incrémental (voir aussi chez l'humain (Piaget, 1967)).

7. Conclusion

Notre contribution. Nous avons présenté notre travail sur la conception de comportements complexes pour des agents autonomes. Ce travail se situe dans un cadre de sélection d'action où un comportement complexe est la combinaison de comportements simples. Notre contribution principale concerne la conception automatique d'un ensemble de comportements de base qui serviront à l'algorithme de combinaison. Cette tâche est essentielle à la réalisation de comportements complexes et est aussi une première étape vers un méta-apprentissage : d'une certaine manière, l'agent est capable de sélectionner les capacités dont il a besoin.

L'agent n'a besoin que des signaux de renforcement et des types d'objets présents dans l'environnement pour définir de manière incrémentale des comportements de plus en plus complexes. La valeur ajoutée d'un nouveau comportement est utilisée pour déterminer s'il viendra compléter l'ensemble des comportements de base de l'agent, de manière à servir à l'avenir pour élaborer des comportements plus complexes. Ce processus itératif se termine quand il n'y a plus de comportements à considérer.

Validation. Notre algorithme a été testé sur le problème du monde des tuiles. Il a fourni de bons résultats puisqu'il a été capable de proposer un ensemble de comportements plus complet que l'ensemble intuitif construit à la main. De plus, il faut noter que les problèmes difficiles auxquels il a été confronté (du fait du nombre d'objets présents) ne peuvent être résolus par de l'A/R classique.

Travaux futurs. Un certain nombre d'améliorations et de problèmes ouverts restent en suspend. Il est très important de travailler à un meilleur critère de sélection des comportements "pertinents", le critère actuel étant une heuristique simple. Tester nos algorithmes sur d'autres problèmes est aussi nécessaire pour déterminer plus précisément ses capacités et mieux comprendre l'influence des quelques paramètres de l'algorithme qui sont toujours réglés manuellement. Enfin, comme l'efficacité de l'algorithme de combinaison peut être affecté par la taille de l'ensemble des comportements de base, ce processus de combinaison mériterait d'être encore étudié et amélioré.

La capacité obtenue ici de répondre à plusieurs motivations simultanées pourrait être des plus utiles au sein d'un système multi-agent coopérant (Jennings *et al.*, 1998). En effet, un agent aura la possibilité d'interagir avec l'un ou l'autre de ses congénères, voire avec plusieurs. On retrouve donc clairement ce besoin de pouvoir faire un compromis entre différents objectifs possibles, et d'apprendre des tâches impliquant des nombres variables de perceptions. Malheureusement, l'apprentissage par renforcement dans un cadre multi-agent est en lui-même déjà un problème des plus complexes (Dutech *et al.*, 2001, Shoham *et al.*, 2003). L'aborder avec l'approche présentée dans cet article serait probablement très ambitieux.

Une perspective intéressante serait que l'agent définisse de lui-même des *abstractions d'objets*. Pour l'instant, les types d'objets dans l'environnement sont fournis à l'agent par un concepteur humain, mais nous pensons que la sélection de comportements pourrait servir à catégoriser automatiquement les objets en classes (telles que "obstacles", "but", "à bouger", etc). Ce serait une nouvelle étape vers un vrai méta-apprentissage et une plus grande autonomie des agents.

8. Bibliographie

Baxter J., Bartlett P., « Infinite-Horizon Policy-Gradient Estimation », *Journal of Artificial Intelligence Research*, vol. 15, p. 319-350, 2001a.

Baxter J., Bartlett P., Weaver L., « Experiments with Infinite-Horizon, Policy-Gradient Estimation », *Journal of Artificial Intelligence Research*, vol. 15, p. 351-381, 2001b.

Bertsekas D., Tsitsiklis J., *Neurodynamic Programming*, Athena Scientific, 1996.

Boutilier C., Reiter R., Price B., « Symbolic Dynamic Programming for First-order MDPs », *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, p. 690-697, 2001.

Buffet O., Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs, PhD thesis, Université Henri Poincaré, Nancy 1, septembre, 2003. Laboratoire Lorrain de recherche en informatique et ses applications (LORIA).

Buffet O., Dutech A., Charpillet F., « Adaptive Combination of Behaviors in an Agent », *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI'02)*, 2002.

Buffet O., Dutech A., Charpillet F., « Automatic Generation of an Agent's Basic Behaviors », *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'03)*, 2003.

Cassandra A. R., Exact and Approximate Algorithms for Partially Observable Markov Decision Processes, PhD thesis, Brown University, Department of Computer Science, Providence, RI, 1998.

Dayan P., Hinton G., « Feudal Reinforcement Learning », *Advances in Neural Information Processing Systems 5 (NIPS'93)*, 1993.

Dietterich T., « Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition », *Journal of Artificial Intelligence Research*, vol. 13, p. 227-303, 2000.

Digney B., « Learning Hierarchical Control Structure for Multiple Tasks and Changing Environments », *Proceedings of the Fifth Conference on the Simulation of Adaptive Behavior (SAB'98)*, 1998.

Dutech A., Buffet O., Charpillet F., « Multi-Agent Systems by Incremental Gradient Reinforcement Learning », *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, 2001.

Dzeroski S., Raedt L. D., Driessens K., « Relational reinforcement learning », *Machine Learning*, vol. 43, p. 7-52, 2001.

Genest C., Zidek J., « Combining Probability Distributions : A Critique and an Annotated Bibliography », *Statistical Science*, vol. 1, n° 1, p. 114-135, February, 1986.

Gretton C., Thiébaux S., « Exploiting First-Order Regression in Inductive Policy Selection », *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI'04)*, 2004.

Hauskretch M., Meuleau N., Kaelbling L., Dean T., Boutilier C., « Hierarchical Solution of Markov Decision Processes Using Macro-Actions », *Proceedings of the Fourteenth International Conference on Uncertainty in Artificial Intelligence (UAI'98)*, p. 220-229, 1998.

Hengst B., « Discovering Hierarchy in Reinforcement Learning with HEXQ », *Proceedings of the Nineteenth International Conference on Machine Learning (ICML'02)*, p. 243-250, 2002.

Humphrys M., « Action Selection methods using Reinforcement Learning », *From Animals to Animats 4 : 4th International Conference on Simulation of Adaptive Behavior (SAB-96)*, September, 1996.

Jaakkola T., Jordan M., Singh S., « On the Convergence of Stochastic Iterative Dynamic Programming Algorithms », *Neural Computation*, vol. 6, n° 6, p. 1186-1201, 1994.

Jennings N., Sycara K., Wooldridge M., « A Roadmap of Agent Research and Development », *Autonomous Agents and Multi-Agent Systems*, vol. 1, p. 7-38, 1998.

Joslin D., Nunes A., Pollack M. E., TileWorld Users' Manual, Technical Report n° TR 93-12, August, 1993.

Kaelbling L., « Hierarchical Learning in Stochastic Domains : Preliminary Results », *Proceedings of the Tenth International Conference on Machine Learning (ICML'93)*, 1993.

Kaelbling L., Littman M., Moore A., « Reinforcement Learning : A Survey », *Journal of Artificial Intelligence Research*, vol. 4, p. 237-285, 1996.

Kearns M., Mansour Y., Ng A., « A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes », *Machine Learning*, vol. 49, p. 193-208, 2002.

Lin. L.-J., « Self-improving reactive agent based on reinforcement learning, planning and teaching. », *Machine Learning*, vol. 8, p. 293-321, 1992.

Lin L.-J., « Hierarchical Learning of Robot Skills », *Proceedings of the IEEE International Conference on Neural Networks (ICNN'93)*, 1993.

Littman M., Cassandra A., Kaelbling L., « Learning policies for partially observable environments : scaling up », *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, 1995.

MacKay D., *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

Mahadevan S., Connell J., « Automatic Programming of Behavior-based Robots using Reinforcement Learning », *Artificial Intelligence*, vol. 55, n° 2-3, p. 311-365, June, 1992.

McCallum R. A., Reinforcement Learning with Selective Perception and Hidden State, PhD thesis, University of Rochester, 1995.

Nehmzow U., Smithers T., McGonigle B., « Increasing Behavioural Repertoire in a Mobile Robot », *From Animals to Animats : Proceedings of the Second Conference on the Simulation of Adaptive Behavior (SAB'93)*, 1993.

Parr R. E., Hierarchical Control and Learning for Markov Decision Processes, PhD thesis, 1998.

Peret L., Garcia F., « On-line search for solving Markov Decision Processes via heuristic sampling », *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'2004)*, 2004.

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

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

Russell S., Norvig P., *Artificial Intelligence : A Modern Approach*, Englewood Cliffs, NJ : prentice Hall, 1995.

Shoham Y., Powers R., Grenager T., Multi-agent reinforcement learning : a critical survey, Technical report, Stanford, 2003.

Singh S., Jaakkola T., Jordan M., « Learning without state estimation in Partially Observable Markovian Decision Processes », *Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, 1994.

Sutton R., Barto G., *Reinforcement Learning : an introduction*, Bradford Book, MIT Press, Cambridge, MA, 1998.

Tyrrell T., Computational Mechanisms for Action Selection, PhD thesis, University of Edinburgh, 1993.

Urzelai J., Floreano D., Dorigo M., Colombetti M., « Incremental Robot Shaping », *Connection Science Journal*, 1998.

Watkins C., Learning from delayed rewards, PhD thesis, King's College of Cambridge, UK., 1989.

Weng J., « A Theory of Mentally Developing Robots », *Proceedings of the 2nd International Conference on Development and Learning (ICDL'02)*, june, 2002.