Développement autonome des comportements de base d'un agent

Alain Dutech[1], Olivier Buffet[1,2], François Charpillet[1]

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

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

Résumé : Nous travaillons à la conception d'agents devant faire face simultanément à plusieurs objectifs, ce qui s'avère difficile, même si chaque objectif est de faible complexité. Le présent article emploie un processus de sélection d'action existant, basé sur des comportements de base (architecture modulaire). Nous y proposons un algorithme de recherche des comportements de base requis via un apprentissage par renforcement incrémental. Cela mène à une architecture très autonome, ne nécessitant que peu de réglages.

Mots-clés : Processus de décision markoviens, apprentissage par renforcement, motivations multiples

1 Introduction

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. Si les éthologues ont étudié ce problème en profondeur, leurs modèles s'avèrent difficiles à implémenter, comme l'explique (Tyrrell, 1993). Un grand nombre de modèles 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. 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* ont été étudiées (voir (Lin., 1992), (Humphrys, 1996) par exemple).

Suivant les principes de (Humphrys, 1996), nous employons une architecture dans laquelle un agent connaît différents comportements de bas niveau (que nous appellons *comportements de base*) et doit choisir une action d'après l'utilité que lui associent ces comportements de base. Les travaux précédents dans ce domaine se sont concentrés sur la sélection de cette action, c'est-à-dire comment combiner diverses quantités représentant les préférences marquées par les comportements de base associés. Alors que ces travaux proposent de fournir à l'agent des comportements définis 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. Pour cela, nous suggérons de donner à l'agent les moyens d'apprendre, de zéro, ces comportements. Ceci signifie non seulement apprendre à accomplir une tâche donnée, mais aussi *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 plus 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.

1.1 Objectifs

Soulignons quelques motivations pour qu'un agent apprenne de lui-même les comportements de base qu'il requiert :

- Rappelons d'abord que la sélection d'action s'attaque à des tâches complexes impliquant des buts parallèles, hétérogènes et parfois concurrents.
- Une méthode pour choisir des comportements de base permettrait de complêter voire remplacer des choix intuitifs de ces comportements.
- L'apprentissage facilite la conception de l'agent (moins de long réglages manuels de paramètres) et accroit grandement son autonomie, puisqu'il peut apprendre ses propres comportements de base.

1.2 Notre cadre de sélection d'action

Les bases de notre architecture de sélection d'action doivent être claires :

- 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 devant répondre à un but/une motivation simple.
- 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.
- Le processus de sélection d'action est de type "flux-libre", comme décrit par (Tyrrell, 1993) : 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* action.

1.3 Exemple

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 1, 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 réellements utiles pour résoudre cette tâche complexe.

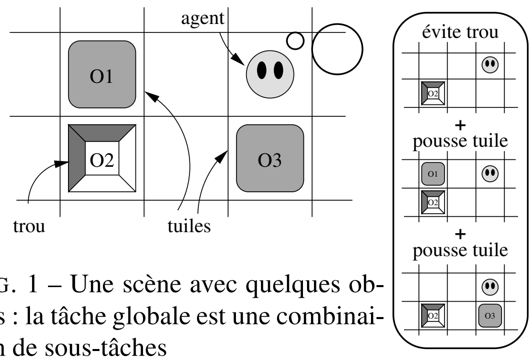


FIG. 1 – Une scène avec quelques objets : la tâche globale est une combinaison de sous-tâches

1.4 Organisation de l'article

Pour présenter notre algorithme de définition automatique de comportements de base, nous introduirons d'abord la notion de comportement de base (section 2). Ensuite la section 3 explique 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

Nous allons voir ici que nos *comportements de base* sont chacun guidés par une *motivation*. Cette section décrit donc ce qu'est une motivation, comment apprendre un comportement, ce que sont les comportements de base, et enfin comment les utiliser.

2.1 Motivation liée à un comportement

Un premier aspect d'une motivation est de savoir quels sont les bons ou mauvais résultats que l'agent devrait essayer d'atteindre ou d'éviter. Pour cela, on va choisir un sous-ensemble parmi l'ensemble des signaux de récompense élémentaires disponibles (deux dans le cas du monde des tuiles : un pour pousser des tuiles dans des trous et un pour éviter des trous). Toutefois, la notion de motivation ne peut se réduire à une telle rémunération. Il faut aussi préciser les types 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 est ainsi décrite par 1- un sous-ensemble de fonctions de récompense élémentaires et 2- un uplet de types d'objets pris en compte (appelé "type de configuration"), ce qui est représenté sur la figure 2. Cette motivation va guider des approches d'apprentissage par renforcement.

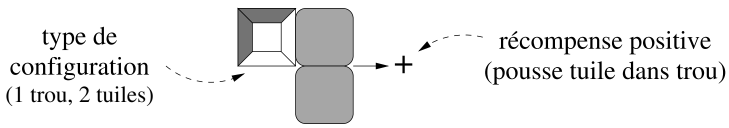


FIG. 2 – Une représentation schématique d'une motivation pour 1 trou, 2 tuiles et une récompense positive $(+)$ quand une tuile est poussée dans un trou.

2.2 Notre cadre d'apprentissage par renforcement

Le cadre général de l'apprentissage par renforcement (A/R) est le suivant (voir (Sutton *et al.*, 1998)). Un agent perçoit son environnement, décide d'une action pour modifier cet environnement, et reçoit en réponse un signal de renforcement. N'utilisant que ce signal (une récompense scalaire) et sans information sur la dynamique de l'environnement, divers algorithmes (tels que le Q-learning (Watkins, 1989), TD(λ) (Tesauro, 1992)) permettent à l'agent d'apprendre automatiquement un comportement (appelé ici politique) pour optimiser la récompense au cours du temps.

Plus formellement, l'A/R peut être exprimé à travers les processus de décision markoviens (MDP) (Puterman, 1994). Des politiques déterministes sont alors parmi les solutions optimales. Formellement, une politique π est une fonction qui, à chaque *état* s du système, associe une action optimale a. Une fonction d'utilité évaluant la "qualité" d'une action est souvent associée avec une politique, et notée $Q(s, a)$.

Dans notre cadre, nous sommes confrontés à une limitation cruciale du cas idéal des MDPs. Notre agent ne percevant son environnement que partiellement, il ne peut connaître l'état complet du système. Face à un processus non-markovien, on fait donc une recherche dans l'espace des politiques stochastiques, lesquelles fournissent des probabilités de choix des actions depuis un certain état (voir (Singh *et al.*, 1994)). Pour un comportement de base donné choisi, une telle politique peut être apprise en employant par exemple un algorithme de descente de gradient (Baxter & Bartlett, 2001; Baxter *et al.*, 2001).

2.3 Comportements de base

Pour une motivation donnée, tout algorithme d'A/R adapté aux observations partielles peut être utilisé. Il n'est requis que : 1- de définir une fonction de récompense "guide" comme la somme de fonctions de récompenses élémentaires sélectionnées, et 2- de construire la perception basée sur les observations des objets (objets correspondants au type de configuration). Le premier résultat obtenu est une politique stochastique. Néanmoins, il faudra emmagasiner d'autres informations pour faire usage de ce “comportement” : le type de configuration servira à trouver les objets adéquats dans l'environnement, et la fonction d'utilité Q aidera à peser l'importance du comportement.

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 décomptée 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** (*bb*)[1] 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.

2.4 Combiner des comportements de base

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 1 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) est associée 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 3.

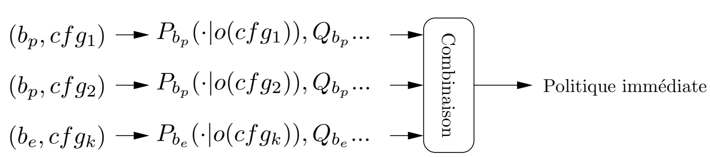


FIG. 3 – Comment obtenir une distribution de probabilité sur les actions avec les trois paires (bb, cfg) identifiées.

[1] *bb* est l'acronyme de “*basic behavior*”.

Note : Les lecteurs intéressés trouverons plus de détails sur la combinaison de comportements en annexe, section A.1.

3 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 : apprendre le meilleur comportement combiné pour une motivation donnée, essayer d'améliorer ce comportement et décider s'il sera alors utilisé comme un nouveau comportement de base.

3.1 Apprendre une combinaison de comportements de base

Etant donné un ensemble de comportements de base $\mathcal{B} = \{bb_1, \cdots, bb_n\}$, un algorithme décrit dans (Buffet *et al.*, 2003) permet de produire une fonction de combinaison efficace de ces comportements. Cette combinaison est basée sur une fonction paramétrée des politiques dans laquelle les paramètres sont appris à l'aide d'un algorithme de recuit simulé. Le comportement résultant de ce processus d'apprentissage est un *comportement combiné* : $cb = \mathcal{F}(\mathcal{B}, \Theta)$ où $\Theta = (\theta_1, \cdots, \theta_n)$ est un vecteur de paramètres appris de la taille de $\mathcal{B}$. La section A.1 dans les annexes donne plus de détails sur l'algorithme de combinaison.

3.2 Amélioration

Une descente de gradient "par renforcement" (Baxter *et al.*, 2001) permet d'améliorer la politique du comportement combiné cb ainsi obtenu, 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.

3.3 Nouveau comportement de base

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. Cette décision est basée sur une comparaison heuristique 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 :

- $|V_{cb} - V_{nb}| > \sigma_s$ (pour éviter des erreurs dues à des estimations proches) et,
- soit $((V_{cb} < 0)$ and $(V_{nb} < \sigma_- * V_{cb}))$ (faible amélioration d'un résultat négatif)
- soit $((V_{cb} \geq 0)$ and $(V_{nb} > \sigma_+ * V_{cb}))$ (amélioration majeure d'un résultat positif).

Ce critère dépend de trois paramètres qui ont une grande influence sur l'algorithme général, comme détaillé en section 4. Dans nos expérimentations, ils ont été réglés manuellement. Un travail important serait de tester leur validité sur d'autres tâches.

4 Algorithme plus détaillé

On cherche ici à automatiser le choix des comportements de base. Comme expliqué en section A, 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.

4.1 Principe

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 les bb déjà sélectionnés pour concevoir les nouveaux. Ceci conduit à construire un ensemble de bb de complexité croissante. Or la définition de la motivation d'un comportement (voir section 2.1) implique que la "complexité" d'un comportement vient de deux sources : la taille du type de configuration, et la combinaison de fonctions de récompense élémentaires.

Pour résumer les principes adoptés, un nouveau type de configuration étant choisi, les comportements correspondants aux possibles combinaisons de récompense sont appris et testés pour déterminer leur intérêt. La figure 4 illustre la progression selon les types de configuration : la complexité croît quand on decend dans l'arbre. Utilisant le processus décrit en section 3.3 pour sélectionner un nouveau comportement de base, la section suivante explique l'algorithme de croissance de l'arbre.

4.2 Croissance de l'arbre

On construit 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 4.1). Chaque nouveau comportement nb est comparé à la combinaison correspondante cb (critère présenté en section 3.3) 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 4, 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).

4.3 Algorithme

Avant de le tester, voyons une définition formelle de notre méthode à travers l'algorithme 1. Parmi les données requises par cet algorithme, on a :

- d_{max} : Comme une croissance sans fin de l'arbre est possible, ce paramètre limite la profondeur de l'arbre.

– d : 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 3.3).

Algorithme 1 Un arbre croissant de comportements de base

Entrées: $Critere(\cdot,\cdot)$: pour évaluer les nouveaux comportements.
d_{max} : profondeur maximale de l'arbre. d : profondeur des sous-arbres développés.

```
T arbre vide. B ensemble vide.
Ajouter à T tous les comportements ayant de 0 à d types d'objets.
pour tout  Comportement b encore à visiter (jusqu'à la profondeur d_max) faire
   Adapter la combinaison des bb courants de B :
   V_cb ← efficacité de ce comportement combiné cb.
   Apprendre b par une recherche directe de politique (initialisée par cb) :
   V_nb ← efficacité de ce nouveau comportement nb.
   si  Critere(V_cb, V_nb) = vrai alors
      Ajouter à T les fils de b ayant jusqu'à d types d'objets en plus.
      Marquer b comme [basique]. L'ajouter à B.
   fin si
   Marquer b comme [visité].
fin pour
```

Sorties: Un ensemble $\mathcal{B}$ de comportements de base retenus.

5 Expérimentations

5.1 Application au monde des tuiles

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 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 1), 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 $\mathcal{B}_{ref}$).

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. 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 1, 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 : $proche(O_3) = vrai$, $direction(O_3) = S$

Paramètres - Les paramètres apparaissant dans la description de l'algorithme 1 de croissance d'arbre (y compris le critère de la section 3.3) ont été réglés manuellement, et définis comme suit : $\sigma_s = 500$ $\sigma_- = 0,9$ $\sigma_+ = 2$ $d_{max} = 4$ $d = 1$

Méthodologie - 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 (dans une grille 8×8). 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). L'efficacité d'une combinaison est mesurée à travers le gain total reçu pendant 100 000 pas de simulation.

5.2 Analyse

Croissance de l'arbre - Observons d'abord l'algorithme de croissance d'arbre : la figure 4 montre l'arbre des comportements évalués qui a été développé. 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. 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 7 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 résoud un autre blocage (figure 7 a) en annexe) 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" ($\sigma_+ = \mathbf{1,3}$), les deux *bb* spécialisés sont inclus dans $\mathcal{B}$, ainsi que 4 autres plus complexes. Cela amène des 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.

Efficacité de l'arbre - Les figures 5 a) et b) 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). Les axes x et y du plan horizontal indiquent le nombre de trous et de tuiles dans les situations

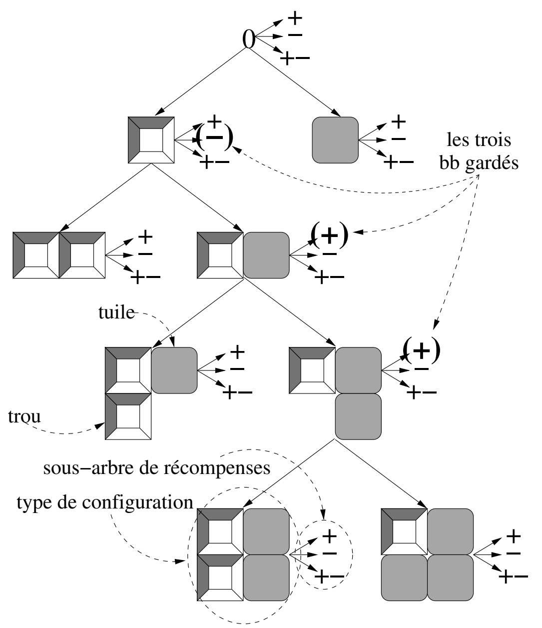
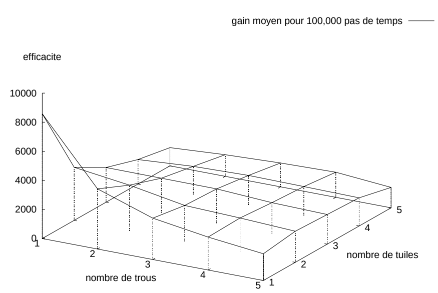
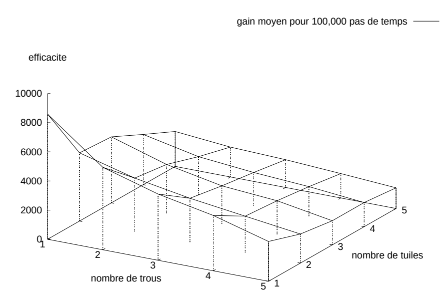


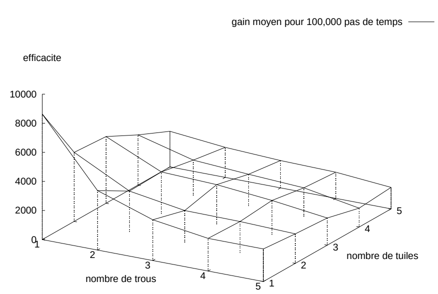
FIG. 4 – L'arbre des comportements testés et (entre parenthèses) ceux qui ont été gardés.



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



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



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

FIG. 5 – Efficacité de la combinaison dans des situations plus complexes avec différentes ensembles de comportements de base.
Note : c) Tests identiques, mais avec $\mathcal{B}'_{arbre}$ = $\mathcal{B}_{arbre}$ + le comportement de base [2-trous/1-tuile avec la récompense "+"].

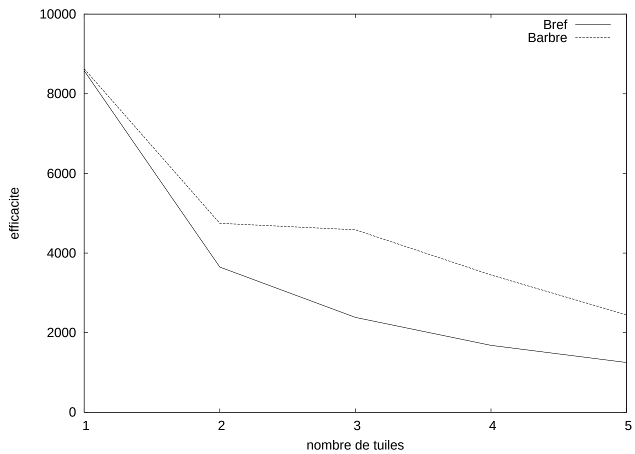


FIG. 6 – Comparaison des efficacités de $\mathcal{B}_{ref}$ et $\mathcal{B}_{arbre}$ avec un trou et plusieurs tuiles.

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 (note : le tableau 1 présente aussi des valeurs numériques pour certains points).

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 $\mathcal{B}_{ref}$ and $\mathcal{B}_{arbre}$ 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 6).

Considérant un nombre croissant de trous, les résultats de la figure 5 b) sont toujours plus proches de la surface de référence. Les variations sont partiellement dues à l'estimation de l'efficacité, et on a vu qu'un problème lié au critère empêche les améliorations dans ces cas à deux trous et plus.

Observation d'un critère plus souple - La figure 5 c) montre les résultats obtenus avec le critère plus souple que nous avons déjà mentionné ($V_{nb} > \mathbf{1,3} * V_{cb}$) et une profondeur maximale de l'arbre $d_{max} = 3$. 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 à 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).

$\#_{trous}/\#_{tuiles}$	$\mathcal{B}_{ref}$	$\mathcal{B}_{arbre}$	$\mathcal{B}'_{arbre}$
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**

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

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 (expérimentations dans (Buffet *et al.*, 2002)). Un agent utilisant l'A/R classique ne réussi qu'à éviter de tomber dans les trous. Mesurer l'efficacité comme sur la figure 5 produirait un pic similaire pour une tuile et un trou, le reste de la surface gisant avec un gain nul. Avec notre méthode, l'agent reste capable de se comporter correctement avec jusqu'à 5 trous et 5 tuiles. Evidemment, il y a un compromis à trouver entre un nombre réduit de comportements de base (apprentissage plus rapide) et des performances élevées.

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 terme de performances. Bien entendu, il faudrait chercher un critère plus pertinent et une façon d'automatiser le réglage des paramètres.

Il serait intéressant de tester et comparer nos travaux sur des projets similaires. Des essais sont actuellement conduits sur le “house environment” conçu et utilisé dans (Humphrys, 1996). C'est un problème plus simple 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 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.

Plus proche de notre travail, (Digney, 1998) propose un algorithme de Q-learning hiérarchique complêtement automatisé qui apprend à créer et structurer des “sous-modules” pour résoudre un problème multi-tâches. Une différence majeure avec notre travail est la perception, ici parfaite, de l'environnement. Les Q-modules appris à l'aide de l'algorithme de Digney semblent plus fortement liés au type de tâches traitées, et il n'est pas clair qu'ils puissent être réutilisés dans des situations très différentes. En outre, son approche de la sélection d'action n'est pas extensible : si de nombreuses sous-tâches sont similaires, Digney doit apprendre autant de Q-modules là où notre algorithme n'aurait besoin que d'un comportement de base. D'un autre côté, le travail de Digney apparaît plus automatisé (moins dépendant de réglages de paramètres) mais toujours restreint à des tâches simples.

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), semblent 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).

Pour toujours considérer des aspects pratiques, notre approche requiert le passage par un environnement simulé, de manière à pouvoir conduire des expérimentations contrôlées. Dans une application sur un problème du monde réel, cela impliquerait de concevoir une telle simulation, par exemple en apprenant un modèle de l'environnement. Cela amène à un autre problème important de l'apprentissage artificiel, mais peut s'avérer plus réaliste qu'une simple approche par essais et erreurs dans un monde réel.

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 sera ajouté à 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 ajouter.

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 ne pourraient être résolu 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.

Une perspective très 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 comportement 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.

Références

BAXTER J. & BARTLETT P. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, **15**, 319–350.

BAXTER J., BARTLETT P. & WEAVER L. (2001). Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, **15**, 351–381.

BUFFET O., DUTECH A. & CHARPILLET F. (2002). Adaptive combination of behaviors in an agent. In *Proceedings of ECAI'02*.

BUFFET O., DUTECH A. & CHARPILLET F. (2003). Automatic generation of an agent's basic behaviors. In *Proceedings of AAMAS'03*.

DIGNEY B. (1998). Learning hierarchical control structure for multiple tasks and changing environments. In *Proceedings of SAB'98*.

HUMPHRYS M. (1996). Action selection methods using reinforcement learning. In *SAB-96*.

JOSLIN D., NUNES A. & POLLACK M. E. (1993). *TileWorld Users' Manual*. Rapport interne TR 93-12.

LIN. L.-J. (1992). Self-improving reactive agent based on reinforcement learning, planing and teaching. *Machine Learning*, **8**, 293–321.

MCCALLUM R. A. (1995). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester.

NEHMZOW U., SMITHERS T. & MCGONIGLE B. (1993). Increasing behavioural repertoire in a mobile robot. In *SAB'93*.

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

SINGH S., JAAKKOLA T. & JORDAN M. (1994). Learning without state estimation in partially observable markovian decision processes. In *Proceedings of ICML'94*.

SUTTON R., PRECUP D. & SINGH S. (1998). *Between MDPs and Semi-MDPs : Learning, planning, and representing knowledge at multiple temporal scales*. Rapport interne, U. of Massachusetts, Dept of Computer and Information Sciences, Amherst, MA.

TESAURO G. (1992). Practical issues in temporal difference learning. *Machine Learning*, (8), 257–277.

TYRRELL T. (1993). *Computational Mechanisms for Action Selection*. PhD thesis, University of Edinburgh.

URZELAI J., FLOREANO D., DORIGO M. & COLOMBETTI M. (1998). Incremental robot shaping. *Connection Science Journal*, **10**.

WATKINS C. (1989). *Learning from delayed rewards.* PhD thesis, King's College of Cambridge.

WENG J. (2002). A theory of mentally developing robots. In *Proceedings of ICDL'02*.

A Annexes

Nous présentons ici brièvement le mécanisme de sélection d'action (apprentissage et combinaison de comportements) utilisé dans nos travaux. Une présentation plus détaillée peut être trouvée dans (Buffet *et al.*, 2002).

A.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 1). 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 associants avec des sous-ensembles d'objets de types appropriés (un tel ensemble est appelé **configuration** (cfg)). Cela amène à une décomposition de la scène en un ensemble $\mathcal{BC}(o)$ de paires $(comportement\ de\ base, configuration)$. Revenant à la figure 1, 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 pour chaque paire ($P_{bb}()$). Et, comme chaque observation est associée avec une observation $o(cfg)$, chaque paire fournit une distribution de probabilités sur les actions $P_{bb}(a|o(cfg))$.

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 probabilités sur les actions, déterminant ainsi la politique réelle immédiate de l'agent.

La combinaison peut prendre des formes diverses. Celle que nous avons employé 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) :

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

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

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**.

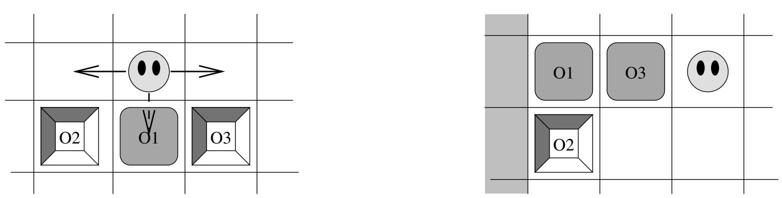
Comme le réglage de ces poids est un problème d'optimisation (maximiser l'espérance du gain moyen), ces paramètres peuvent être appris par renforcement à l'aide d'un recuit simulé. L'algorithme a juste à faire une bonne estimation de l'espérance du gain moyen pour chaque jeu de paramètres.

On notera que, dans une certaine mesure, le processus complet de sélection d'action est "extensible" puisque plusieurs instances d'un même *bb* peuvent être utilisées sans avoir à apprendre de nouveaux paramètres.

A.2 Problèmes ouverts

L'algorithme que nous venons de brièvement présenter a été testé sur le monde des tuiles dans (Buffet *et al.*, 2002) 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. Toutefois, notre méthode (section 4.1) permet de raffiner la politique obtenue par simple combinaison.
- **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 sont requis pour avoir un agent adaptable, mais d'autres, plus spécialisés, permettraient de répondre à des cas difficiles précis (par exemple les deux blocages de la figure 7). C'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.

FIG. 7 – 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.