HAL open science

Apprentissage par Renforcement Développemental en Robotique Autonome

Luc Sarzyniec, Olivier Buffet, Alain Dutech

To cite this version:

Luc Sarzyniec, Olivier Buffet, Alain Dutech. Apprentissage par Renforcement Développemental en Robotique Autonome. Conférence d'Apprentissage - CAP 2011, May 2011, Chambéry, France. inria-00633426

HAL Id: inria-00633426

https://inria.hal.science/inria-00633426v1

Submitted on 18 Oct 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Apprentissage par Renforcement Développemental en Robotique Autonome

Luc Sarzyniec(1), Olivier Buffet(1), Alain Dutech(1,2)
(1) Equipe MaIA, LORIA, INRIA
Campus Scientifique, BP 239, 54506 Vandoeuvre les Nancy, France
`prenom.nom-@-loria.fr`
(2) Auteur à contacter

31 mars 2011

Résumé

Cet article présente une approche développementale de l'apprentissage par renforcement dans un cadre de robotique autonome. Le but est de permettre au robot de tirer parti de la richesse de son environnement sans que cette richesse ne le noie sous trop d'information à traiter. L'idée principale testée ici est de faire croître les capacités perceptives et motrices de l'agent au fur et à mesure de l'apprentissage. Cette approche est combinée avec un apprentissage par renforcement classique (*Q-Learning*) s'appuyant sur un approximateur de fonction non-linéaire (perceptron multi-couche). Au travers d'expériences encore préliminaires, nous montrons le potentiel de cette approche.

Mots-clef Approche Développemental, Apprentissage par Renforcement, Robotique, Apprentissage Multi-Tâches et Transfert

1 Introduction

Le cadre de l'apprentissage par renforcement (Sutton & Barto, 1998) est particulièrement séduisant du fait qu'il permet à un agent artificiel d'apprendre un plan d'action pour résoudre un problème qui peut être mal connu ou incertain et ce en n'utilisant qu'un signal scalaire pour distinguer certaines bonnes ou mauvaises situations. L'apprentissage n'est pas supervisé et il n'est pas nécessaire qu'un expert ou un oracle soit disponible. De plus, le formalisme des processus décisionnels de Markov (Puterman, 1994) permet d'appréhender les aspects théoriques de nombreux algorithmes, garantissant, par exemple, l'existence d'une solution au problème d'apprentissage posé et la convergence vers cette solution optimale.

Bien que pouvant s'enorgueillir de succès applicatifs certains (backgammon, contrôle d'une flottille d'ascenseurs, pilotage d'un hélicoptère, *etc.*), l'utilisation

pratique des outils de l'apprentissage par renforcement pour des problèmes réalistes n'est pas chose aisée. Les difficultés viennent généralement de la taille des problèmes à résoudre qui ne peuvent être décrit qu'à travers un très grand nombre d'états, ce qui rend alors computationnellement irréaliste l'utilisation des algorithmes d'apprentissage par renforcement recherchant une solution exacte. Des algorithmes cherchant des solutions approchées existent (Itération sur les valeurs avec approximation, LSPI, recherche directe de politique par montée de gradient, *etc*, voir les chapitres 1 et 3 du tome 2 du livre commun du Groupe PDMIA (2008)), mais ils sont loin d'apporter des réponses à toutes les difficultés rencontrées en pratique.

Nous nous intéressons particulièrement à l'utilisation de l'apprentissage par renforcement dans le cadre de la robotique autonome. Plusieurs difficultés y réduisent l'applicabilité des algorithmes d'apprentissage. Les données issues de capteurs sont à valeur continues, souvent bruitées, coûteuses à obtenir (en temps et en énergie) et très (trop) riches en information (par exemple dans le cas d'un flux vidéo). Il est certes possible et classique de pré-traiter ces données mais deux situations peuvent se présenter. D'un côté, le pré-traitement peut être trop spécifique, offrant alors une représentation de l'espace perceptif d'une taille certes limitée mais trop spécifiques et par trop *ad hoc* : le problème à apprendre y est trivial. D'un autre côté, une représentation plus générale court le risque d'être d'une taille déraisonnable, rendant la tâche d'apprentissage trop difficile. Le compromis entre ces deux extrêmes est difficile à trouver et ces pré-traitements limitent de fait l'autonomie du robot qui reste très dépendant du concepteur du robot.

Dans cet article, nous proposons une méthode d'apprentissage par renforcement qui tire parti des avantages liés aux deux cas que nous venons d'évoquer : apprentissage facilité car s'appuyant sur un espace perceptif de taille réduite tout en ayant la possibilité de s'appuyer, à terme, sur un espace perceptif aussi complexe que nécessaire et ne limitant donc pas l'autonomie du robot. L'idée est de faire évoluer la taille de l'espace perceptif du robot au cours de la tâche d'apprentissage. Plus le robot devient performant, plus les représentations sur lesquelles il s'appuie pour prendre ses décisions deviennent complexes. Cette démarche s'étend aussi, et dans le même temps, aux capacités motrices du robot ainsi qu'à la difficulté du comportement que doit apprendre le robot. Nous qualifions cet apprentissage de "*développemental*" car il s'inspire des mécanismes de développement présents chez les jeunes enfants et identifiés, notamment, par la psychologie du développement et les neurosciences cognitives. Les recherches montrent comment les mécanismes d'apprentissage profitent de cette prise de conscience progressive de la complexité de son corps et de son environnement par l'enfant (Turkewitz & Kenny, 1985; Kuhl, 1999).

Dans un premier temps, cet article présente les concepts clefs de l'apprentissage par renforcement avant de détailler les difficultés liées à son utilisation dans le cadre de la robotique autonome. Ensuite, nous exposons les grandes lignes de la méthode d'apprentissage par renforcement développementale que nous proposons. Les détails de la mise en œuvre de cette méthode pour réaliser des expérimentations sur un robot de type Khepera-III forment le cœur de la

section 5. Nous discutons les résultats obtenus lors de ces expérimentations dans la section 6 et mettons ce travail en perspective avant de conclure.

2 Apprentissage par renforcement

Le cadre de l'apprentissage par renforcement (AR) s'appuie sur l'hypothèse que l'on peut modéliser le problème par un processus décisionnel de Markov (MDP) (Bellman, 1957). Un MDP se compose d'un n-uplet $(\mathcal{S}, \mathcal{A}, p(), r())$ où $\mathcal{S}$ est un ensemble d'états, $\mathcal{A}$ un ensemble d'actions, $p(s'|s,a)$ décrit la dynamique du processus par une probabilité de transiter de l'état s à l'état s' sous l'effet de l'action a et $r(s,a)$ indique la récompense instantanée reçue en choisissant cette action a alors que le processus était dans l'état s. Un MDP admet au moins une solution optimale sous la forme d'une *politique* $\pi^* : \mathcal{S} \longrightarrow \mathcal{A}$ qui indique, pour chaque état, quelle est la meilleure action à effectuer pour maximiser, sur le long terme, une certaine fonction de la récompense (typiquement $\sum_{t=0}^{\infty} \gamma^t r_t$ où $\gamma \in [0;1[$ est un facteur de pondération et r_t est la récompense reçue après la t^{eme} action choisie).

Sous certaines hypothèses dont nous reparlerons en section 3, les algorithmes d'apprentissage par renforcement permettent de trouver une telle solution optimale π^*, même quand le modèle du MDP (c'est-à-dire les fonctions $p()$ et $r()$) n'est pas connues par l'agent apprenant (Sutton & Barto, 1998). Dans ce travail, nous nous inspirons principalement de l'algorithme du `Q-Learning` (Watkins, 1989) qui vise à apprendre la fonction de valeur optimale Q^* sur les couples (état,action), fonction qui vérifie l'équation d'optimalité de Bellman, à savoir :

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} Q^*(s',a'). \tag{1}$$

Une fois cette fonction apprise, la politique optimale s'en déduit immédiatement par :

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s,a). \tag{2}$$

3 Difficultés de l'AR en robotique

Parmi les hypothèses nécessaires pour garantir la convergence des algorithmes d'apprentissage par renforcement vers une solution optimale figure la nécessité pour l'agent apprenant de visiter un nombre infini de fois chaque couple (état,action) du processus. Même en se contentant d'un très grand nombre de visites, cette contrainte est l'une des principales limitations à l'utilisation des techniques d'apprentissage par renforcement dans le cadre de la robotique. Nous en détaillons quelques raisons.

a) **Espace d'état continu**. L'espace perceptif d'un robot est généralement continu et, bien que le formalisme des MDP reste valide dans ce contexte, les algorithmes d'apprentissage nécessitent classiquement un espace d'état

discret et fini. Si une discrétisation de l'espace d'état est possible, elle est généralement mal adaptée et soit trop fruste, soit computationnellement ingérable car trop détaillée. Une solution est de rechercher une approximation de la fonction de valeur.

b) **Bruit de perception**. Du fait de la nature même des capteurs équipant un robot, les perceptions de ce dernier sont bruitées. Formellement, cela revient à dire que le robot ne dispose pas de la donnée de l'*état* du processus mais d'une simple *observation* imprécise de cet état. Les rares algorithmes d'apprentissage par renforcement existant pour cette extension des MDP au cas partiellement observable ne peuvent traiter que des problème où la taille de l'espace d'état est très petite et donnent des solutions généralement très sous-optimales (Dutech & Samuelides, 2003).

c) **Echantillons coûteux**. Obtenir un échantillon (s, a, s, r), c'est-à-dire effectuer une action a dans un contexte perceptif s pour en observer le résultat, en terme de nouvel état s' et d'une éventuelle récompense r, n'est pas anodin en robotique. Au minimum, cela demande un certain temps et, dès lors, il devient impossible de générer des corpus d'apprentissage conséquent. C'est un problème d'autant plus crucial que le *Q-Learning* ne s'appuie que sur des interactions effectives entre l'agent et son environnement. Des techniques adaptées à cette problématique où l'on ne dispose que de peu de données existent, notamment en mémorisant et en réutilisant plusieurs fois les mêmes données, mais là encore le problème reste largement ouvert (Riedmiller, 2005).

d) **Richesse de l'environnement**. Ce qui pourrait sembler un avantage est aussi, et surtout, une difficulté pour les algorithmes d'apprentissage par renforcement. L'environnement du robot étant riche et varié, il regorge d'informations que le robot peut prendre en compte pour décider de son action. En conséquence, la dimension de l'espace d'état peut croître de manière disproportionnée et, finalement, mettre à mal tout algorithme d'apprentissage. On retrouve le "*Frame Problem*" énoncé par (McCarthy & Hayes, 1969) qui reste un problème majeur dès lors qu'on ne veut pas limiter artificiellement – et souvent de manière *ad hoc* – l'espace perceptif de l'agent, ce qui revient à terme à l'empêcher d'augmenter son autonomie et limiter ses capacités. Cet aspect du problème de l'apprentissage est au cœur de nos préoccupations.

4 Notre approche : *Q-Learning* développemental

Le cœur de notre approche s'appuie sur une approche développementale de l'apprentissage par renforcement. Nous précisons ce concept avant de montrer comment il s'intègre avec une approche neuronale de l'algorithme du *Q-Learning*.

4.1 Approche développementale

Parmi les nombreux concepts et approches mis en avant dans le cadre de la robotique développementale (Lungarella *et al.*, 2003), nous voulons explorer plus avant les apports possibles d'un accompagnement progressif de l'agent apprenant en terme de complexité de la tâche d'apprentissage. Cet accompagnement prend principalement la forme d'une augmentation progressive de la richesse des perceptions et des actions potentielles de l'agent au fur et à mesure que ses performances dans la tâche à apprendre progressent. De même, la difficulté de la tâche elle-même peut être amenée à augmenter au fur et à mesure de l'apprentissage. Le but principal de cette approche est de permettre à l'agent apprenant de tirer le maximum de la richesse et de la complexité du monde sans être pénalisé par cette trop grande richesse, répondant ainsi à la problématique d) exposée en 3

Dans cette optique, plusieurs problèmes se présentent. Il faut d'une part que l'architecture d'apprentissage mise en place permettent de gérer de manière la plus transparente (du point de vue de l'agent) et la plus autonome possible ces augmentations progressive de perception, motricité et difficulté de la tâche. D'autre part, nous voulons que l'agent puisse s'appuyer sur ce qu'il a déjà appris pour apprendre dans un environnement plus riche. Cette dernière problématique est particulièrement délicate, elle se rapproche de la problématique du transfert de connaissance, domaine de recherche encore largement ouvert (Caruana, 1997).

4.2 *Q-Learning* neuronal

Comme nous l'avons dit, notre approche d'apprentissage par renforcement développemental s'inspire largement de l'algorithme du *Q-Learning* et cherche donc à estimer la fonction de valeur optimale. Le principe de l'algorithme est d'utiliser chaque échantillon d'apprentissage (s_t, a_t, s_{t+1}, r_t) fournit à l'instant t par une interaction de l'agent avec son environnement pour mettre à jour l'estimation de la fonction de valeur pour le couple (s_t, a_t). L'algorithme s'appuie sur l'équation de Bellman (1) de la manière suivante :

$$Q^*_{t+1}(s_t, a_t) \longleftarrow (1-\alpha)Q^*_t(s_t, a_t) + \alpha\left(r + \gamma \max_{a' \in \mathcal{A}} Q^*_t(s_{t+1}, a')\right), \qquad (3)$$

où α est un coefficient d'apprentissage, qui peut dépendre de s, de a et du temps.

Pour gérer l'espace perceptif continu, nous utilisons un approximateur de fonction non-linéaire sous la forme d'un *perceptron multi-couche*. Ainsi, pour chaque action a de l'ensemble des actions (que nous considérons comme un espace discret, fini et de taille raisonnable), un perceptron multi-couche est utilisé pour estimer la fonction $Q^*(s, a)$, continue en s.

Pour pallier la faible quantité des échantillons d'apprentissage, nous utilisons les concepts de *traces d'éligibilités* et de *ré-utilisation* des échantillons. L'utilisation de traces d'éligibilité permet d'employer plus efficacement un échantillon d'apprentissage (s_t, a_t, s_{t+1}, r_t) afin de ré-estimer la fonction de valeur Q^* non seulement pour le couple (s_t, a_t) mais aussi pour tous les couples d'état-action

précédents $\{(s_i, a_i)\}_{i<t}$ en pondérant cette mise à jour par un facteur d'atténuation exponentiellement dégressif λ^{t-i}, avec $\lambda \in [0, 1[$. Les traces d'éligibilités et la réutilisation des échantillons accélèrent en général l'estimation de la fonction de valeur, mais il n'y a plus de garantie théorique de convergence vers la fonction de valeur optimale (Bertsekas & Tsitsiklis, 1996; Szepesvári, 2010).

La réutilisation des échantillons passés est un concept beaucoup plus simple. Il s'agit de "rejouer" des interactions passées que l'agent a mémorisées et, ainsi, une même interaction peut potentiellement servir un nombre infini de fois pour apprendre. Les limites d'une telle approche est principalement liée au fait que l'environnement sensori-moteur de l'agent n'est pas exploré et utilisé d'une manière uniformément aléatoire, ce qui, une fois de plus, ne permet pas de garantir la convergence vers une solution optimale. Cette limitation est d'autant plus importante que les échantillons peuvent de plus être "spatialement" mals répartis dans l'espace sensori-moteur, problème aussi abordé par (Baranes & Oudeyer, 2010; Ménard & Frezza-Buet, 2005).

Ces modifications et extensions à l'algorithme du *Q-Learning* répondent principalement aux limites a) et c) formulées précédemment (voir section 3). Cela forme une base sur laquelle nous nous appuyons pour proposer une approche développementale de l'apprentissage par renforcement, que nous détaillons maintenant et qui vise à répondre à ces limites mais aussi à la limite d) concernant la richesse de l'environnement.

4.3 Adéquation avec l'approche développementale

L'architecture d'apprentissage à base neuronale que nous utilisons permet aisément de mettre en œuvre l'approche développementale que nous voulons expérimenter. L'espace d'état étant continu, une partie de sa richesse est liée à la dimension de cet espace. Ainsi, augmenter cette richesse signifie pour nous augmenter la dimension de l'espace d'état, ce qui revient à augmenter le nombre de neurones de la couche d'entrée de notre approximateur neuronal. De même, l'espace d'action étant discret, sa richesse est liée, entre autres, à sa cardinalité et en augmenter la richesse revient alors à utiliser plus de réseaux de neurones (un réseau par action, cf section 4.2).

Que l'on augmente la taille de la couche d'entrée d'un réseau ou le nombre de réseaux, se pose alors le problème de l'initialisation des poids des nouvelles connections créées. Le but est d'essayer de réutiliser au mieux les compétences déjà acquises. Au lieu de tirer aléatoirement leur valeur, notre approche utilise les valeurs des poids de connexion de neurones "sémantiquement" proches pour initialiser ces nouveaux poids (voir le détail en section 5.2). Le but est ainsi de profiter de l'expérience déjà acquise dans des contextes moins riches pour faciliter l'apprentissage dans un contexte plus riche.

5 Expérimentations

5.1 Plateforme expérimentale

Nos expériences robotiques sont effectuées sur un robot KheperaIII de l'entreprise K-Team [1]. Bien qu'il soit doté d'un processeur, nous avons préféré déporter les traitements et les calculs sur une machine externe, les communications entre cette machine et le robot se font par wifi.

Le robot se déplace à l'aide de deux roues motrices qui lui permettent de tourner sur place. Nous avons utilisé 5 actions discrètes : `tourner-droite`, `tourner-gauche`, `stop`, `tourner-droite-lentement` et `tourner -gauche-lentement`. Parmi les capteurs du robot, nous n'utilisons que les capteurs de distance infrarouge pour stopper le robot s'il s'approche trop d'un obstacle. Le capteur principal que nous utilisons est une caméra wifi placée au dessus du robot (voir figure 1) et qui permet de capturer une image couleur 320x200 au format bitmap, image qui est ensuite traitée pour fournir une information sensorielle évolutive dans un espace de dimension moindre.

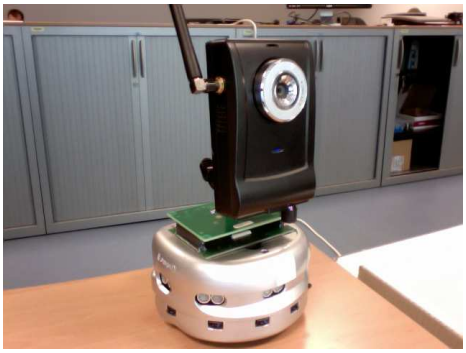

FIGURE 1 – Le robot Khepera III et sa caméra.

Ainsi, par le biais d'un capteur logique que nous appelons la "*rétine*", nous fournissons au robot une perception constituée d'un vecteur de dimension $\dim^{\text{per}}$ où chaque coefficient du vecteur, compris entre 0 et 1, correspond au taux d'une teinte particulière calculée sur une bande verticale de l'image. Le nombre et la position de ces bandes verticales est paramétrable. Sur l'exemple de la partie droite de la figure 2, le robot fait face à un losange bleu sur fond rouge, la rétine est configurée pour donner les taux de bleu sur des bandes verticales – nommées `B1`, `B2` et `B3` – positionnées aux abscisses 80, 160 et 240 de l'image. Les perceptions du robot dans ce cas sont donc $[0,42; 0,67; 0,33]$.

Dans les expériences menées, le robot est placé dans un environnement fermé. Les murs qui l'entourent sont de couleur rouge et des cibles bleues, de formes et de tailles variables, peuvent être placées sur les murs.

1. Voir `http://www.k-team.com`.

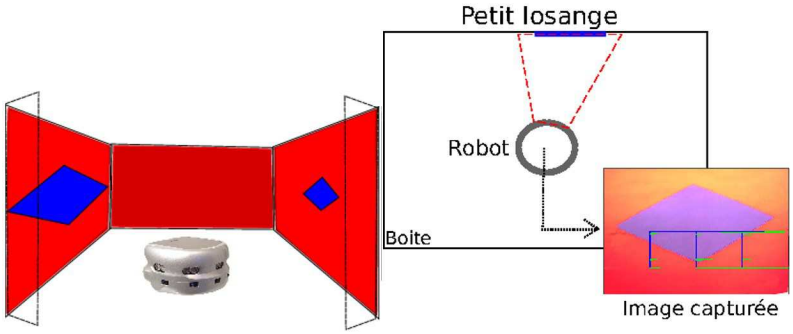

FIGURE 2 – Gauche : le robot dans son environnement. Droite : les perceptions données par la `rétine` du robot. Pour chacune des bandes verticales, en positions $(80, 160, 240)$, le taux de bleu est fourni, soit $[0,42; 0,67; 0,33]$.

5.2 Architecture et algorithme d'apprentissage par renforcement

Notre algorithme s'inspire du *Q-Learning* et essaie d'estimer la fonction de valeur optimale $Q(s, a)$. L'espace perceptif étant continu, nous utilisons un approximateur non-linéaire (perceptron multi-couche) pour chaque action, ainsi que l'illustre la figure 3. La taille de la couche d'entrée des perceptrons est égale à la dimension du vecteur de perception fourni par la `rétine`, soit $\dim^{\text{per}}$. Après une interaction (s_t, a_t, s_{t+1}, r_t), les poids du perceptron de l'action a_t sont modifiés en utilisant l'algorithme de rétropropagation du gradient avec, comme sortie cible du perceptron, une estimation de la fonction de valeur calculée selon l'équation (3).

Quand les perceptions se font plus complexes, c'est-à-dire que la taille du vecteur de perception augmente, on ajoute des neurones d'entrée à chaque perceptron (figure 4, gauche). Quand c'est le nombre d'actions possibles qui augmente, on ajoute autant de perceptrons que de nouvelles actions (figure 4, droite). Les poids des connexions nouvellement créées sont initialisés en copiant les poids des connexions sémantiquement proches. Par exemple, quand l'action `tourner-gauche-lentement` est ajoutée, le perceptron créé est une copie du perceptron associé à l'action `tourner-gauche`. Ensuite, avec les nouvelles interactions entre l'agent et son environnement, les connexions de chacun des deux perceptrons vont évoluer de manière indépendante.

5.3 Expériences

Les expériences visent essentiellement à valider le concept d'une augmentation progressive de la richesse des capacités du robot pour faciliter l'apprentissage.

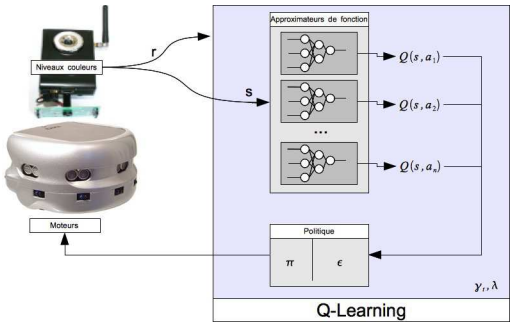

FIGURE 3 – L'architecture interne du robot. Pour chaque action a, un perceptron multi-couche avec autant de neurones d'entrée que de dimensions $\dim^{\text{per}}$ dans le vecteur de perception s est utilisé pour estimer $Q(s,a)$. A partir de ces estimations, une politique ϵ-gourmande permet de décider de l'action à effectuer.

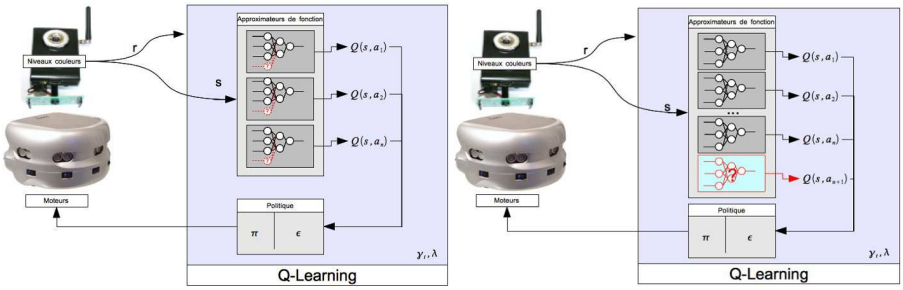

FIGURE 4 – Augmentation des capacités sensori-motrices du robot. A gauche, quand on augmente la richesse des perceptions, on ajoute de nouveaux neurones en entrée des perceptrons. A droite, quand on augmente la richesse des actions, on ajoute de nouveaux perceptrons.

Pour cette expérience, le robot est placé dans un environnement fermé, relativement petit, constitué de 4 murs où deux cibles bleues, de taille différentes, sont présentes (voir figure 2, partie gauche). Le robot perçoit cet environnement à l'aide d'une rétine qui fournit un vecteur perceptif de dimension 3, les bandes verticales étant positionnées en $(80, 160, 240)$. Le robot reçoit une récompense de $+1$ quand il est fait face à une des cibles. En pratique, si $\vec{\mathbf{s}}$ est le vecteur de perception, le robot est récompensé quand

$$\vec{\mathbf{s}}.[0.7; 1.0; 0.7]^\top \geq 1,6.$$

Dès lors, la plage d'obtention d'une récompense est plus large lorsqu'il est face à la grande cible que lorsqu'il fait face à la petite cible. Par le biais d'une politique d'exploration aléatoire uniforme, en partant de plusieurs positions initiales, le robot génère un corpus d'apprentissage d'environs 2.000 exemples, en utilisant ses 5 actions.

Dans l'approche *développementale*, le robot apprend d'abord en n'utilisant que 3 actions (il ne peut pas tourner lentement). Cet apprentissage se déroule pendant 100.000 pas de temps, chaque pas de temps utilisant un échantillon *compatible* du corpus d'apprentissage en terme d'action. Un exemple d'une politique de bonne qualité apprise au terme de cette phase se trouve en figure 5. Dans un deuxième temps, on permet au robot d'utiliser les 5 actions. Les perceptrons multi-couche correspondant à ces nouvelles actions sont initialisé en copiant le perceptron multi-couche de l'action rapide correspondante (ainsi, le perceptron de l'action `tourner-gauche-lentement` est la copie du perceptron de `tourner-gauche`). De nouveau, le robot apprend pendant 100.000 itérations, en utilisant tout le corpus d'apprentissage. Un exemple de comportement appris à l'issue de cette deuxième phase est montré figure 6.

L'approche développementale est comparée avec une approche *directe* où le robot dispose dès le début de ses 5 actions et apprend pendant 200.000 pas de temps, en utilisant tout le corpus d'apprentissage. Cette approche semble aussi permettre d'apprendre des politiques de bonne qualité, comme le montre la figure 7.

Pour mieux comparer les deux approches, nous avons utilisé une évaluation plus quantitative de la qualité des politiques apprises. Tout les 25.000 itérations d'apprentissage, le robot est placé dans son environnement et dispose de 100 actions pour accumuler le plus de récompense possible. Lors de cette évaluation, il est périodiquement déplacé "à la main" dans une position aléatoire. La figure 7 montre l'évolution de cette performance pour les deux approches suivies. On y voit en particulier que l'ajout des 2 nouvelles actions fait temporairement chuter la qualité de la politique de l'approche développementale. Par contre, le politique finale atteinte avec la méthode dévelopementale est meilleure que celle de l'approche directe. On peut d'ailleurs noter que la qualité de la politique développementale est toujours supérieure à celle de la politique directe. Ces résultats, issus d'une expérience, restent préliminaires et montrent aussi à quel point l'utilisation d'approximateur neuronaux pour la fonction Q est délicate et rend l'apprentissage non-monotone (nous n'expliquons pas autrement l'inflexion de la courbe directe vers 150.000 itérations).

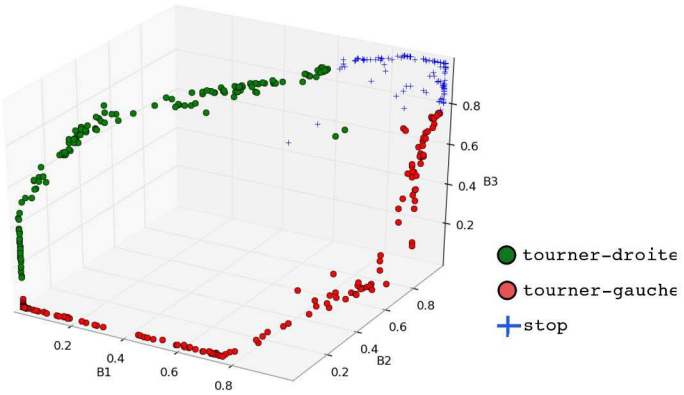

FIGURE 5 – Politique apprise avec 3 bandes visuelles en entrée et 3 actions, après 100.000 itérations. Chaque point correspond à une des perceptions du corpus d'exemple dans l'espace perceptif à 3 dimensions.

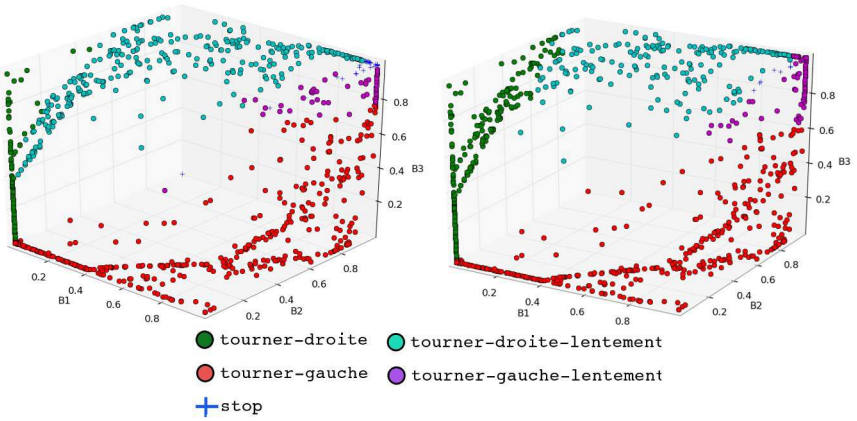

FIGURE 6 – Politiques après apprentissage. A gauche : Approche *Développementale*, à droite : Approche *Directe*. Chaque point correspond à une des perceptions du corpus d'exemple dans l'espace perceptif à 3 dimensions.

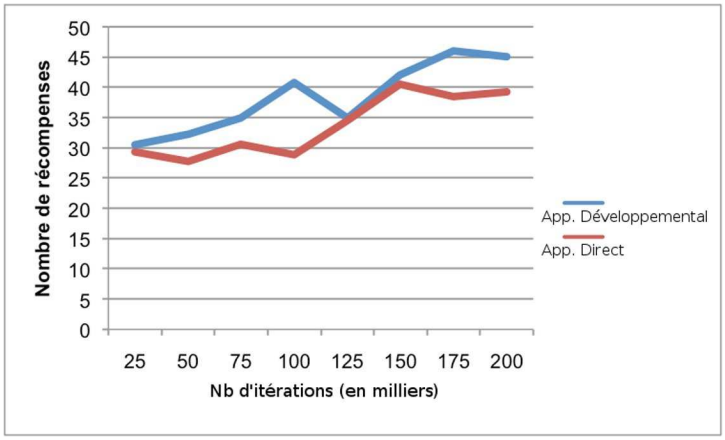

FIGURE 7 – Comparaison de l'apprentissage direct et développemental. Au cours de l'apprentissage toutes les 25.000 itérations, la performance des politique apprises est testée en laissant le robot agir "au mieux" en fonction de sa politique courante. La performance est la somme des récompenses reçues pendant 100 itérations.

Au cours de ces expériences, nous avons expérimenté plusieurs valeurs pour les paramètres λ et α mais γ valait toujours 0.9. Les courbes présentées ont été réalisés avec $\alpha = 0,01$, quant à λ il variait *au cours d'une expérience* de la façon suivante :

Nb itérations	0	50.000	100.000	125.000
App. Développemental	0,6	0	0,6	0
App. Direct	0,6	0,6	0	0

6 Discussion et Perspectives

Nos travaux ouvrent de large perspectives dans plusieurs directions. Les résultats obtenus sont encore très préliminaires et doivent être consolidés, non seulement dans le cadre de l'augmentation de la richesse des actions mais aussi, comme nous sommes actuellement en train de l'expérimenter, dans le sens de l'augmentation de la richesse de perception.

Le fait d'utiliser un perceptron multi-couches et la réutilisation des données d'apprentissage ("*replay*") nous a été inspiré par l'algorithme d'apprentissage *Neuro Fitted Q Iteration* de Riedmiller (2005) qui est efficace dans le cas où l'on dispose de peu de données. Nous avons ajouté l'utilisation de traces d'éligibilité pour accélérer l'apprentissage. Pour autant, l'apprentissage est loin d'être aussi rapide et efficace que nous l'avions espéré : il faut non seulement plusieurs millier d'exemples mais aussi et surtout plusieurs centaines de milliers d'itérations pour apprendre. Nous envisageons de continuer à explorer l'influence des paramètres

d'apprentissage, d'utiliser des règles d'apprentissage plus performante pour les perceptrons multi-couches mais aussi d'autres schémas d'approximations, plus simple comme les *plus proches voisins*, ou plus évoluées comme l'approche de (Quinton & Buisson, 2008).

Une autre approche, plus différente, est de ne plus chercher à apprendre la fonction de valeur Q. Comme l'erreur d'estimation de Q qui guide l'apprentissage est issue d'une équation de point fixe, la convergence de l'apprentissage est lente car l'estimation de l'erreur en un point dépend de la qualité des estimations de tous les perceptrons. Une possibilité est de chercher directement à apprendre une politique sans en calculer sa valeur, comme dans les travaux de (Baxter & Bartlett, 2001). Ces autres schémas d'apprentissage par renforcement ne sont pas forcément directement compatibles avec l'approche développementale que nous voulons expérimenter et certaines s'avèreront peut-être sans intérêt dans notre cadre.

Nos expérimentations ne nous ont pas permis non plus de nous confronter à une difficulté liée à l'approche développementale que nous n'avons pas encore détaillé ici. Il s'agit de problèmes qui peuvent apparaître lors de l'enchaînement dans l'apprentissage de tâches. Par exemple, le robot peut apprendre d'abord à s'approcher d'une cible (pour l'identifier) puis, dans un deuxième temps, apprendre à partir à droite ou à gauche selon la forme de la cible. Mais rien n'assure qu'une fois légèrement éloigné de la cible, le robot n'y revienne pas, surtout si l'espace perceptif n'est pas assez riche pour différencier les deux situations. Outre le fait d'augmenter la richesse de perception, il semble aussi possible de jouer avec le ou les signaux de renforcement, mais tout ceci reste à concevoir et à tester.

7 Conclusion

Cet article a présenté une approche développementale de l'apprentissage par renforcement qui s'appuie sur un enrichissement progressif des capacités d'un robot durant l'apprentissage. Cette approche a pour but de permettre à un agent de tirer parti de la richesse de son environnement sans être écrasé par cette richesse. Nous avons combiné un approximateur non-linéaire (perceptron multi-couche) et un algorithme d'apprentissage par renforcement classique (*Q-Learning*) pour expérimenter cette approche développementale sur un robot. La tâche reste encore trop simple, les expériences n'ont fait que défricher le problème mais les perspectives ouvertes sont nombreuses et prometteuses. Nous pensons étendre très rapidement nos tests au cas où la dimensions perceptive s'enrichit pour ensuite explorer le problème plus ouvert de l'apprentissage d'enchaînement de comportements.

Références

Baranes A. & Oudeyer P.-Y. (2010). Maturationally-constrained competence-based intrinsically motivated learning. In *Proc. of IEEE Int. Conference on Devel-*

opment and Learning (ICDL 2010), Ann Arbor, Michigan, USA.

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

Bellman R. (1957). *Dynamic programming.* Princeton University Press, Princeton, New-Jersey.

Bertsekas D. & Tsitsiklis J. (1996). *Neuro-dynamic programming.* Athena Scientific, Belmont, MA.

Caruana R. (1997). *Learning to Learn*, chapter Multitask Learning. Kluwer Academic Publishers.

Dutech A. & Samuelides M. (2003). Apprentissage par renforcement pour les processus décisionnels de Markov partiellement observés. *Revue d'Intelligence Artificielle (RIA)*, **17(4)**, 559–589.

Groupe PDMIA (2008). *Processus Décisionnels de Markov en Intelligence Artificielle. (Edité par Olivier Buffet et Olivier Sigaud)*, volume 1 & 2. Lavoisier - Hermes Science Publications.

Kuhl P. (1999). *The New Cognitive Neurosciences*, chapter 8 - Language, Mind, and Brain : Experience Alters Perception, p. 99–115. Bradford Books.

Lungarella M., Metta G., Pfeifer R. & Sandini G. (2003). Developmental robotics : a survey. *Connection Science*, **15**(4), 151–190.

McCarthy J. & Hayes P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, **4**, 463–502.

Ménard O. & Frezza-Buet H. (2005). Model of multi-modal cortical processing : Coherent learning in self-organizing modules. *Neural Networks*, **18**(5-6), 646–655.

Puterman M. (1994). *Markov Decision Processes : discrete stochastic dynamic programming.* John Wiley & Sons, Inc. New York, NY.

Quinton J.-C. & Buisson J.-C. (2008). Interactivist sensorimotor learning : computational implementation and parallel optimization. In *Proceedings of The 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (Nice, France).*

Riedmiller M. (2005). Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method. In J. a. Gama, R. Camacho, P. Brazdil, A. Jorge & L. Torgo, Eds., *Machine Learning : ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*, chapter 32, p. 317–328. Berlin, Heidelberg : Springer Berlin / Heidelberg.

Sutton R. & Barto A. (1998). *Reinforcement Learning.* Bradford Book, MIT Press, Cambridge, MA.

Szepesvári C. (2010). *Algorithms for Reinforcement Learning (Synthesis Lectures on Artificial Intelligence and Machine Learning).* Morgan and Claypool.

Turkewitz G. & Kenny P. (1985). The role of developmental limitations of sensory input on sensory/perceptual organization. *Developmental & Behavioral Pediatrics*, **6**(5), 302–306.

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