Utilisation d'aide pour l'apprentissage par renforcement

Fabien Montagne* Philippe Preux† Samuel Delepoulle*

7 avril 2004

Résumé

Dans cette contribution, on s'intéresse au problème de la fourniture d'une aide à un apprenant par renforcement. Pour cela, nous proposons un schéma algorithmique ayant pour objet de répondre à ce problème, l'architecture critique-critique. Le cœur de cette architecture est un agent dont le comportement est régi par un algorithme d'apprentissage par renforcement de type acteur-critique. Celui-ci interagit avec son environnement dans lequel se trouve une entité qui lui donne des conseils. Ce schéma algorithmique est ouvert et nous en discutons différents points.

1 Introduction

Dans la vie réelle, l'apprentissage est souvent accéléré par l'intervention d'un agent extérieur qui donne des exemples ou montre comment réaliser certaines tâches, ou comment obtenir certains résultats. Cette intervention peut prendre différentes formes, parmi lesquelles :

- démonstration du comportement : un agent (virtuel ou non) montre le comportement que l'apprenant doit émettre [Lin92, Sch97, HD01] ;
- imitation : l'apprenant observe un agent et utilise ses observations pour améliorer son comportement [BK96, GMBQ97, BD99, PB03] ;
- guidage : on « prend l'apprenant par la main » pour lui faire « ressentir » le comportement à réaliser [Pom93, RB02] ;
- instruction : un agent donne des explications / directives décrivant le comportement à émettre, ses conditions d'émissions [MS96, Clo97]. Une version extrême de cette approche consiste à programmer l'agent explicitement avec les éléments de comportement attendu ;
- façonnage : ce qui consiste à modifier le problème de décision résolu par l'agent pour en simplifier la résolution (donc, augmenter sa probabilité de résolution), de manière à ce que

*Université du Littoral Côte d'Opale, Laboratoire d'Informatique du Littoral, Calais, France, montagne@lil.univ-littoral.fr, delepoulle@lil.univ-littoral.fr

†Université de Lille 3, Groupe de recherche en Apprentissage Automatique, Villeneuve d'Ascq, France, philippe.preux@univ-lille3.fr

le comportement obtenu soit correct pour le problème initial. Différentes approches sont possibles : modifier la fonction retour, modifier la fonction de transitions, modifier l'espace d'états, d'actions, ... [DC94, Mat94, Gul97, SRT98, RA98, Del00].

Dans la suite, nous nous concentrons sur l'approche consistant à fournir des indications (sous une forme ou une autre) concernant le comportement à émettre à une entité adaptative, de manière à ce que l'intervention soit la plus légère et non contraignante possible (pour l'humain). Conceptuellement, cela signifie combiner l'apprentissage par renforcement avec l'apprentissage supervisé (à base d'exemples).

Ci-dessous, nous recensons quelques points importants qui seront étudiés dans la suite. Le schéma algorithmique que nous proposons apporte des réponses à ces trois problèmes couplés.

Représenter et utiliser de l'aide

Tout en utilisant des algorithmes d'apprentissage par renforcement, notre but est d'aider son apprentissage en lui fournissant des indications qui vont l'aider à avoir un comportement de bonne qualité. L'apprenant acquiert ses connaissances de deux manières : par l'aide reçue et par interaction avec son environnement. L'utilisation de telles techniques soulève de nouveaux problèmes tels que :

- l'obtention des exemples pour aider l'agent : on aimerait que ces exemples soient les moins coûteux à produire et les plus utiles pour l'apprenant. Aussi, nous considérons dans la suite que l'aide fournie n'est ni complète, ni parfaite. Elle n'est pas complète dans le sens où elle n'indique par une trajectoire depuis un état initial à un état final. Elle n'est pas parfaite dans le sens où elle n'est pas optimale ; elle peut être plus proche d'un optimum local que de l'optimum global (si tant est que ces notions d'optimalité aient un sens) ; cependant, on suppose que la trajectoire n'est pas mauvaise ;
- le représentation et le stockage de l'aide pour maximiser la généralisation à partir de cette aide ;
- l'intégration incrémentale de l'aide dans l'apprenant au fil de son fonctionnement ;
- la combinaison de cette aide avec l'apprentissage réalisé par l'apprenant en interagissant avec son environnement.

Les trois premiers points sont communs avec l'apprentissage supervisé, même si ici, le contexte est sensiblement différent.

L'espace d'états

Lors de la mise en pratique d'algorithmes d'apprentissage par renforcement, l'un des points critiques est la description de l'espace d'états. Outre le choix des caractéristiques de l'environnement qui seront représentées dans l'état de l'apprenant, il se pose ici le problème de la représentation de l'aide dans un espace d'états. Nous expliquons plus loin qu'il nous semble intéressant d'utiliser deux espaces d'états, l'un pour l'apprenant pour représenter sa police, l'autre pour représenter l'aide.

La sélection de l'action

De plus, les algorithmes d'apprentissage par renforcement posent le problème de la sélection de l'action à réaliser afin d'optimiser l'équilibre en exploration et exploitation. En particulier, quand l'apprenant dispose d'une aide, il se pose la question de savoir jusqu'où suivre cette aide (l'exploiter), quand et comment la remettre en cause.

1.1 Un survol de notre proposition

Représenter et utiliser de l'aide

Nous proposons une architecture d'agent apprenant par renforcement s'appuyant sur l'architecture acteur-critique et l'algorithme du TD(λ) [SB98]. Notre architecture est constituée de deux critiques et d'un acteur. Chaque critique fournit une information concernant la valeur de l'état courant ; ces deux informations sont combinées et leur combinaison est transmise à l'acteur qui l'utilise pour décider de l'action à effectuer. Le premier critique apprend à partir de l'aide fournie ; le second critique apprend lors de l'interaction avec l'environnement (*cf.* fig. 2).

L'information fournie par chacun des critiques peut être diverse. En particulier, nous envisageons trois possibilités :

- chaque critique fournit une estimation de la valeur d'un état ;
- l'un des critiques peut fournir une estimation de la valeur d'un état tandis que le second fournit une correction de cette estimation ;
- l'un des critiques peut fournir une estimation de la valeur d'un état tandis que l'autre fournit une confiance dans cette estimation. Cette troisième solution rappelle l'idée de carte de compétence de Thrun & Möller [TM92].

Si on note $A(\Phi(s))$ l'estimation de la valeur de l'état issue de l'aide et $C(s)$ l'estimation de l'état courant faite par le second critique, alors dans les trois cas cités ci-avant, la valeur de l'état courant $V(s)$ est le résultat de l'équation $V(s) = A(\Phi(s))$ op $C(s)$.

L'espace d'états

Les deux critiques peuvent utiliser une représentation distincte de l'espace d'états. En particulier, le premier critique (utilisant l'aide fournie) peut utiliser une représentation grossière de l'espace d'états alors que le second (apprenant par interaction) utilise une description complète des états. L'idée sous-jacente est que l'aide fournie est grossière (comme on l'a vu plus haut) et qu'il faut donc la représenter de manière grossière, sans s'attacher à des détails non pertinents dus à l'expression approchée de l'aide. Autrement dit, il ne faut pas que la représentation de l'aide soit plus précise que ce qu'elle est. Hors, pour agir, il est possible qu'une représentation plus fine du contrôle de l'agent soit nécessaire, d'où l'utilisation de deux représentations différentes des états. On peut d'ailleurs imaginer utiliser plus de deux représentations (*cf.* la discussion).

Pour illustrer cette idée, nous utilisons plus loin le problème de l'apprentissage d'une trajectoire optimale par un mobile sur un circuit. Typiquement, l'état est représenté, notamment, par une position et une vitesse. Pour aider l'agent à effectuer une trajectoire optimale dans un virage en suivant la corde, la position est plus importante que la vitesse : aussi, si l'on ne tient pas compte de la vitesse dans la représentation de l'état utilisée par le premier critique, on généralise immédiatement l'apprentissage.

Donc, on note :

- $\mathcal{S}$ l'espace d'états « complet » utilisé par le second critique (apprenant par interaction) : il utilise donc une représentation fine des états et on le nomme « critique fin » CF ;
- Σ l'espace d'états « grossier » utilisé par le premier critique (apprenant à partir de l'aide) : utilisant une représentation grossière de l'état, on le nomme « critique grossier » CG ;
- $\Phi : \mathcal{S} \mapsto \Sigma$: une fonction de projection du premier dans le second. Φ est une surjection.

À partir de l'aide, CG apprend une estimation de la valeur $A(\sigma)$, pour une partie de l'ensemble des états $\sigma \in \Sigma$. Pour sa part, CF apprend une correction ou une confiance $C(s)$ sur l'ensemble des états $s \in \mathcal{S}$.Il est a noter que même si pour CF la tâche à réaliser est un problème de décision markovien, l'utilisation d'un sous-espace d'état peut nous placer au niveau du CG dans un POMDP.

La sélection de l'action

L'utilisation d'un guide a pour but de minimiser l'exploration faite par CF sans pour autant limiter ses possibilités. Ainsi aidé, il est possible pour l'agent CF d'éviter une partie de la phase d'exploration. En cas de succès, il exploite les conseils du CG adaptés à son environnement. Et ce n'est qu'en cas d'échec qu'il devra explorer son environnement. De plus, même en cas d'échec cela n'implique pas que l'aide de CG soit mauvaise ; il est possible que ce soit la partie de l'état s non décrite dans σ qui soit à remettre en cause. L'agent CF va donc explorer cette partie de l'espace d'états avant d'invalider l'aide procurée par CF. Si on considère l'exemple simple d'un suivi de trajectoire, l'aide peut être une suite de position donnée σ dans un espace d'états discrets à forte granularité, le second critique évoluant dans un espace d'états plus fin. Ainsi sur la figure 1, on voit que si CG n'apporte qu'une aide grossière, il nous permet d'explorer l'environnement sur une faible partie de l'espace d'états [1]

Plan de la suite

Dans la suite de cette contribution, nous commençons par préciser l'architecture critique-critique et l'algorithme d'apprentissage associé. Ensuite, nous réalisons une étude expérimentale sur le problème de la recherche d'une trajectoire optimale d'un mobile sur un circuit. Enfin, nous discutons des problèmes et questions en suspens.

[1] Sur ce type d'exemples, l'optimalité de la trajectoire trouvée n'est pas garantie, mais un bon choix de l'espace d'état CG peut apporter des solutions.

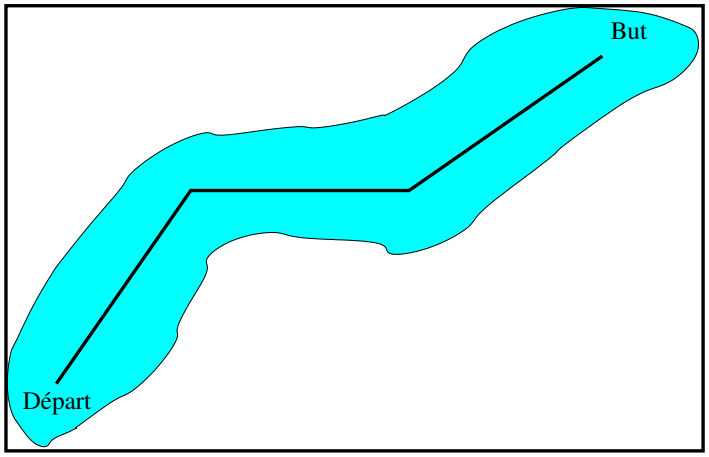


FIG. 1 – Exemple de suivi de trajectoire. Si le CG donne les conseils représentés par la ligne noire, le CF a la possibilité de ne pas explorer l'ensemble de son espace d'états (qui pourrait être l'ensemble du plan représenté) mais uniquement la zone bleue.

2 L'architecture critique-critique

L'utilisation d'aide dans l'apprentissage par renforcement a pour but d'assister l'agent dans la sélection des actions. Pour cela, le CG transmet à l'agent apprenant par renforcement (CF) une estimation de la qualité[2] de l'état ($A(\Phi(s))$). Le CF va ensuite confronter cette estimation avec son expérience et en modulant sa propre évaluation ($C(s)$), corriger cette estimation (voir Figure 2).

Néanmoins, en mode supervisé, il est plus difficile de manipuler une description aussi fine d'un état qu'en mode d'apprentissage par renforcement. On utilise généralement des algorithmes d'apprentissage par renforcement pour des problèmes que l'on ne sait pas résoudre facilement de façon supervisé (soit insoluble, soit coût de calcul trop élevé). Ceci écarte les solutions de guidage que ce soit par le biais d'un acteur ou l'indication de valeur d'états. Nous utilisons deux niveaux de représentations des états : l'un utilisant une représentation partielle et l'autre plus fine.

Afin de pouvoir extraire des informations facilement exploitables par l'agent apprenant par renforcement, l'agent d'aide utilise une description de l'état suffisamment précise pour être une aide pour l'agent apprenant par renforcement, mais aussi suffisamment large pour généraliser son apprentissage à base de l'aide.

À partir de l'architecture acteur-critique, nous proposons de réaliser cette idée avec une architecture où l'acteur reçoit en complément un second critique, soit un critique pour chaque niveau de description des états. Le critique à la vue grossière (CG) donne une estimation de la qualité de l'état que l'on notera $A(s)$. De son côté le critique qui a une description plus fine (CF) associe une correction, confiance, à l'estimation que l'on notera $C(s)$. Dans les premiers épisodes, le critique

[2] on utilise ici le mot « qualité » dans un sens non conventionnel par rapport à l'apprentissage par renforcement ; on l'utilise dans son sens intuitif.

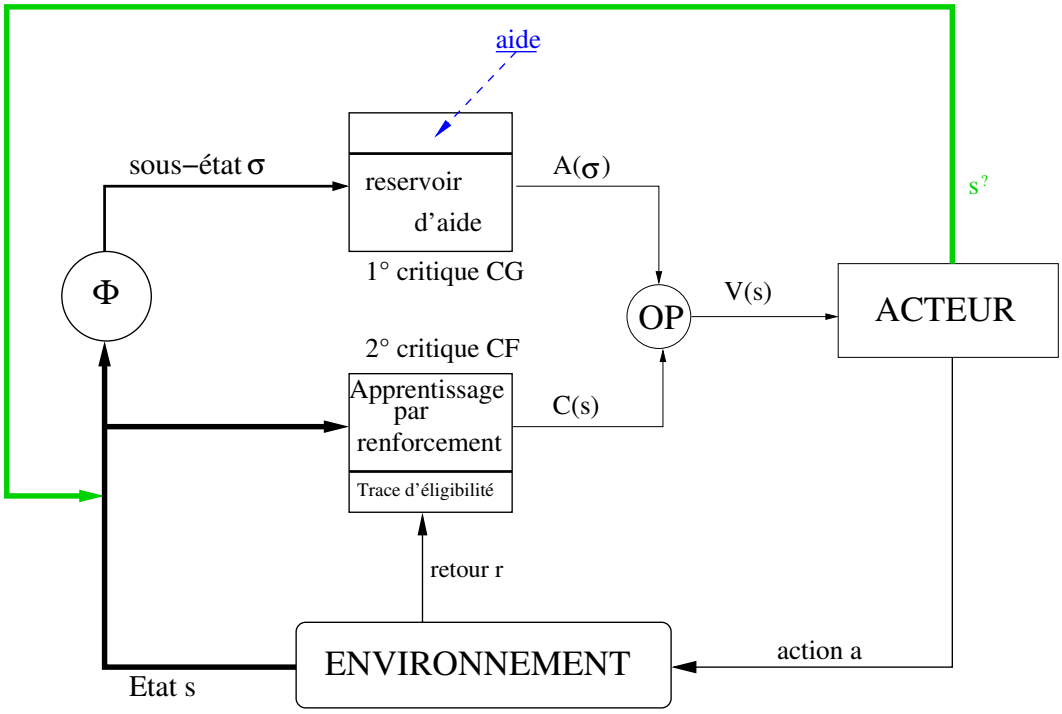


FIG. 2 – L'architecture critique-critique : les deux critiques utilisent leur vision de l'environnement pour donner chacun une valeur associée à l'état, lors de la sélection de l'action, c'est une combinaison de ces deux valeurs qui sera utilisée. L'environnement est perçu par l'acteur dans un état s. Lors de la sélection de l'action, l'acteur recherche les états accessibles depuis l'état courants. Il envoie ensuite ces états requêtes (notés $s^?$) vers les deux critiques afin de déterminer leur valeur. Une fois le choix de l'état cible fait, l'acteur émet l'action correspondante, l'environnement passe donc dans un nouvel état et le second critique CF reçoit un retour r correspondant.

CF n'ayant aucune expérience, l'acteur sera plus influencé dans son choix de l'action par l'avis de l'agent CG.

3 Algorithme Critique-critique

3.1 Traduction de l'aide sous forme numérique

La façon la plus simple d'utiliser l'aide de CG dans l'architecture critique-critique est de traduire cette aide sous forme de grandeurs numériques. La traduction de l'aide sous une forme numérique est très dépendante à la fois de la tâche à résoudre mais aussi de la formulation de l'aide. Aussi, c'est lors de la formalisation de la tâche à résoudre, qu'il faudra définir comment traduire l'aide en grandeur numérique. Une solution est de construire un gradient de qualité des états en fonction des exemples : plus un état σ_t est proche d'un état terminal σ_T, plus sa valeur est proche de celle de l'état σ_T.

3.2 Utilisation conjointe d'aide et par d'apprentissage par renforcement

Les deux critiques CG et CF fournissent chacun une information sur la valeur de l'état courant que l'on note respectivement :

- $C(s)$: confiance dans, ou correction de l'information $A(\Phi(s))$,
- $A(\Phi(s))$: gradient obtenu à partir de l'aide.

La valeur d'un état s est obtenue par combinaison de ces deux quantitées $V(s) = C(s)$ op $A(\Phi(s))$. On a choisi d'étudier deux modes de combinaisons, le produit et l'addition. L'utilisation de l'un ou de l'autre opérateur ne remet pas en question le rôle de CG. Par contre, il modifie la façon dont CF intervient dans l'apprentissage.

Le produit : cet opérateur nous permet de considérer l'évaluation de l'état par CF comme étant un niveau de confiance envers l'estimation de cet état par CG. En diminuant la confiance, c'est-à-dire $C(s)$, on diminue aussi la valeur $V(s)$. Ainsi, cet opérateur permet de remettre en question l'estimation de la valeur de l'état fournie par le CG. Son point faible est l'apprentissage de nouvelles politiques. En effet, si l'estimation de l'état par le CG est très basse, il sera difficile pour CF de faire croître la valeur de $V(s)$. Une solution possible est de faire en sorte que le CG adapte lui aussi ses estimations en fonction des expériences. On pourra en particulier considérer une phase d'apprentissage supplémentaire pour le CG (cette phase d'apprentissage serait une sorte de bilan des épisodes passés et donc se ferait de temps en temps).

L'addition : l'opérateur d'addition permet lui aussi de remettre en cause les estimations de la valeur des états par CG, mais dans ce cas, plutôt que de parler de confiance, le rôle de l'agent apprenant par renforcement CF est celui de correcteur. L'opérateur d'addition permet plus facilement de faire émerger de nouvelles solutions que l'opérateur produit. En effet, $\forall s, s' \in \mathscr{S}$ tels que $A(\Phi(s)) < A(\Phi(s'))$ et s, s' sont accessibles depuis un état s_0 et si s est meilleur que s', alors on peut trouver une valeur de $C(s)$ telle que $V(s) > V(s'), \forall C(s')$. Pour le produit, si par exemple $A(\Phi(s)) = 0$ et $V(s') > 0$, pour favoriser l'état s, il ne suffit pas d'augmenter $C(s)$, il faut faire diminuer la valeur $V(s')$ c'est-à-dire diminuer $C(s')$.
En conclusion les deux opérateurs considérés là définissent un rôle différent pour CF. Même si le produit peut nécessiter une phase d'apprentissage supervisée et hors-ligne pour extraire une politique, il permet assez rapidement de déduire la qualité de l'aide en chaque état rencontré et le cas échéant explorer son environnement. D'autre part, l'opérateur d'addition semble plus direct dans son apprentissage en permettant facilement l'extraction de politique. Mais cet opérateur nécessite de considérer des espaces de valeurs plus larges ce qui influe sur la fonction de retour à définir. Enfin d'autres possibilités pour combiner $C(s)$ et $A(\Phi(s))$ sont envisageables.

3.3 Remise en cause de l'aide, apprentissage en ligne

L'algorithme d'apprentissage utilisé pour l'agent apprenant par renforcement est inspiré du TD(λ). Le choix de la fonction retour est difficile à déterminer. Le rôle de CF étant de corriger ou modérer l'avis du CG, la fonction de renforcement peut être assez simple. Afin d'aider l'architecture CC à explorer son environnement, en plus de la fonction de retour, on lui adjoint une fonction de pénalisation. Son but est de s'attaquer aux problèmes de boucles qui peuvent se créer : l'agent ne fait plus de différence entre différents états voisins et choisi soit de rester dans un état s non terminal, soit d'appliquer une politique dans laquelle il boucle sur un certain nombre d'états. En dévaluant d'une valeur α la valeur $C(s_t)$ chaque état s_t rencontré et à chaque rencontre, on diminue l'estimation de la valeur V de cet état, ainsi on augmente la probabilité d'exploration de l'entourage proche de l'état s_t. Lors de la distribution de la récompense suivant la trace d'éligibilité, chaque état verra sa valeur augmentée de α. Cette fonction doit être séparée de la fonction de retour car elle ne dépends pas de la position temporelle de l'état dans la politique suivie, elle dépend uniquement de la fréquence de rencontre de chaque état suivant la politique choisie.

C'est en respectant ces différentes contraintes qu'a été établi l'algorithme critique-critique (Cf. algo 1).

4 Expérimentations

Afin de tester notre architecture critique-critique et l'algorithme du même nom, l'exemple de support est le circuit automobile.

4.1 Présentation du problème

Le circuit automobile est un exemple classique en apprentissage par renforcement, Il s'agit de faire évoluer un mobile sur un circuit (voir Figure 3), le mobile ayant pour tâche d'apprendre à faire le tour du circuit en un temps minimal sans sortir de la piste.

La mise en œuvre de l'architecture critique-critique sur cet exemple se fait aisément. En effet, guider l'agent dans ce cas consiste à lui donner une succession de points à atteindre, ou plus simplement lui donner une notion d'ordre sur les points à atteindre. Néanmoins, cela ne suffit pas à déterminer une trajectoire optimale car s'il est facile de déterminer l'ensemble des points à atteindre, il est plus complexe de donner les vitesses instantanées associées à chaque point. D'où l'utilisation de deux niveaux de lecture d'un même état :

- pour l'agent d'aide CG, un état σ_t est décrit par la position $P(x, y)$ du mobile à l'instant t ;
- pour l'agent apprenant par renforcement CF, la description d'un état s_t a pour composante la position du mobile $P(x, y)$ à l'instant t enrichi de la vitesse instantanée du mobile. Cette vitesse est contrainte : une vitesse limite a été définie, et même si l'agent continue d'accélérer, sa vitesse restera égale à la vitesse limite. D'autres éléments pourraient être ajoutés à la

Algorithme 1 *Algorithme du critique-critique* : η est le taux d'apprentissage, γ le taux de dépréciation, λ le coefficient habituel du TD(λ), $e(s)$ la trace d'éligibilité de l'état s. Voir le texte pour les autres notations. Remarquons que η peut être distingué pour chaque état (donc $\eta(s)$) comme cela est requis pour la convergence de l'algorithme.

// 1- **Initialisation de l'apprentissage**
Apprentissage par CG de l'aide : ici, des couples ($\Phi(s)$,A($\Phi(s)$))
$\forall s$ initialisation de C(s)
// 2- **Apprentissage non-supervisé**
répéter
  s_0=*état initial*
  $\forall s, e(s) = 0$
  // 3- Suivi d'une trajectoire
  tant-que s_t non terminal **faire**
    $\mathscr{S}$=s, s atteignable depuis s_t à l'instant $t+1$
    pour $s \in \mathscr{S}$ **faire**
      $\sigma = \Phi(s)$
      V(s)=C(s) op A(σ)
    fin pour
    s'=$\underset{s\in\mathscr{S}}{\text{argmax}}$(V($s$))
    si s' n'est pas dans la trace d'éligibilité **alors**
      Ajouter s' à la trace d'éligibilité
    fin si
    Dévaluer la valeur de l'état s : $C(s) = C(s) - \alpha$, $0 < \alpha < 1$, α constante
    $\delta = r + \gamma C(s') - C(s)$
    $C(s) = C(s) + \eta(r + \gamma C(s'))$
    $e(s) = 1$
    pour tous les s de la trace d'éligibilité **faire**
      $C(s) = \eta\delta e(s) + C(s)$
      $e(s) = \lambda\gamma e(s)$
    fin pour
  fin tant-que
  pour tous les s de la trace d'éligibilité **faire**
    Augmenter la valeur l'état s : $C(s) = C(s) + \alpha$
  fin pour
jusque tous les épisodes sont terminés

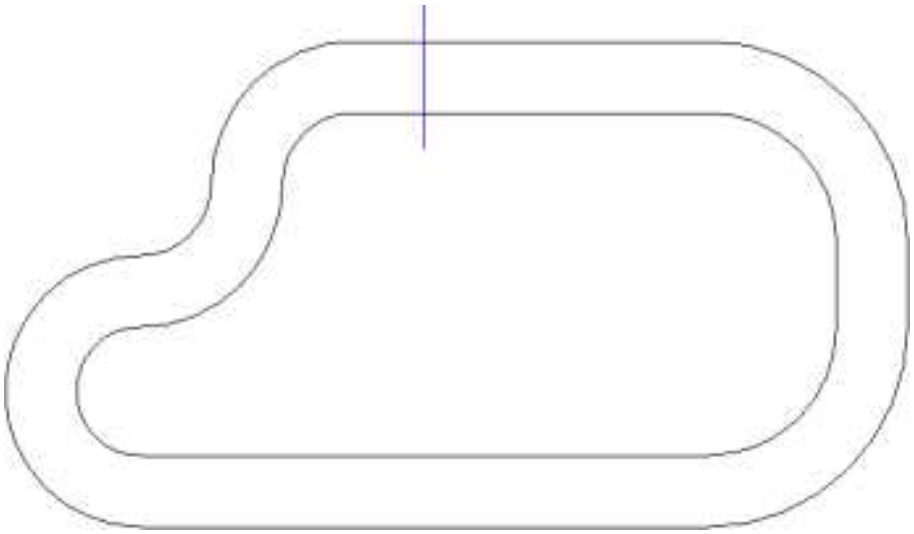


FIG. 3 – Le circuit exemple : le but de notre agent est d'apprendre à terminer un tour de circuit sans sortir de la piste.

description de l'état tels que la quantité de carburant, l'usure des pneumatiques...

On se place ici dans un espace discret. L'espace des actions est formé d'un ensemble d'accélérations. Ces accélérations sont bornées. Ainsi, si à un instant t le mobile a atteint sa vitesse maximale (en fin de ligne droite par exemple), il ne peut s'arrêter à l'instant $t+1$. Ceci complexifie la tâche d'apprentissage, d'où la nécessité d'inclure la vitesse au niveau de l'agent CF. De plus, lors de la mise en place du gradient de qualité $C(\Phi(s))$, l'accélération n'avais pas été prise en compte. L'acteur en suivant l'avis du CG ne prends pas compte de la vitesse, or cette dernière est une composante importante de la fonction de transitions entre deux états successifs. Dans ce cas, l'acteur se place donc dans un POMDP.

Dans le problème du circuit automobile, le but est triple : il faut que l'agent reste sur la piste, qu'il fasse le tour du circuit, mais en un minimum de temps. La fonction la plus simple permettant de résoudre ce problème est la fonction suivante :

$$f(S_t) = \begin{cases} -1 & \text{si l'agent sort de la piste,} \\ 1 & \text{si l'agent à atteint la ligne d'arrivée,} \\ 0 & \text{sinon.} \end{cases}$$

Sans une aide, un agent utilisant une telle fonction, ne réussi pas cette tâche : au fil des sorties du circuit, l'agent apprend vite à rester sur place. Ceci est du au fait que même si un retour plus important est possible en atteignant le bout de la piste, ce gain est trop difficile à atteindre.

4.2 Détails de l'agent stockant l'aide

Au niveau de l'implantation, le CG est simulé par une fonction qui retourne une aide $A(\sigma) \in [0, 1]$ qui est fonction de la zone dans laquelle se trouve le mobile (la première zone est la première ligne droite, la seconde est le premier virage...) Cf.Fig.4. L'utilisation de cette fonction aide permettrait de travailler avec un espace d'états $\sum$ qui est continu.

Pour définir cette fonction valeur, le circuit a été découpé en dix zones distinctes tel que décrit sur la figure 4. Pour chaque zone, la différence de potentiel $(A(s) - A(s')$ pour deux états s et $s')$ entre les extrémités de zone est de 0.1, le CG ne donne donc pas une trajectoire précise à suivre mais plus une idée générale de la trajectoire à suivre. Ainsi, dans la zone 4, quels que soient les états s et s' dont les coordonnées sont respectivement $(\mathbf{x}, y)$ et $(\mathbf{x}, y')$ leurs valeurs sont identiques car, dans l'espace d'états Σ, les distances avec les extrémités sont identiques. De plus, la figure 5 fait apparaître que la quantité retournée par l'aide dépend de la distance parcourue, mais que cette dépendance n'est pas linéaire.

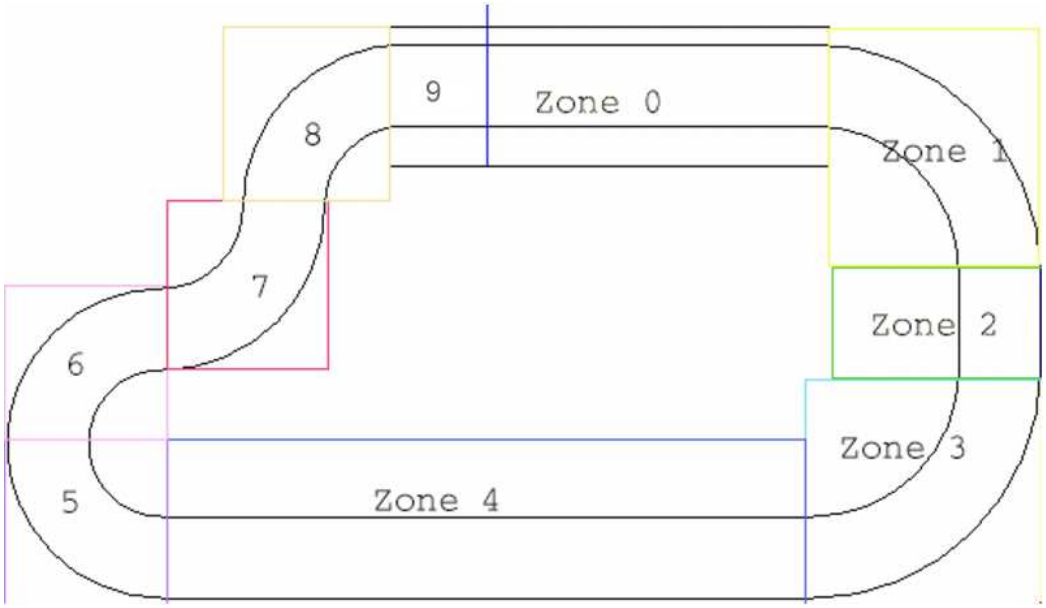


FIG. 4 – Les dix zones définissants la fonction d'aide. Pour chaque zone, pour deux états s en début de zone et s' en fin de zone, $A(s) - A(s') = 0.1$ quelque soit la longueur de la zone.

En complément, en cas de sortie de piste, la fonction d'aide retourne 0 et, si le tour est achevé, elle retourne 2.

4.3 L'agent CF apprenant par renforcement

Pour l'agent apprenant, l'algorithme utilisé est un TD(λ) tabulaire. Ceci implique une discrétisation de l'espace d'états. Utilisé dans une version naïve, le TD(λ) tabulaire nous contraint à enregistrer $5.85 \cdot 10^6$ états. Une version utilisant un réseau de neurones à déjà été étudiée mais des problèmes de

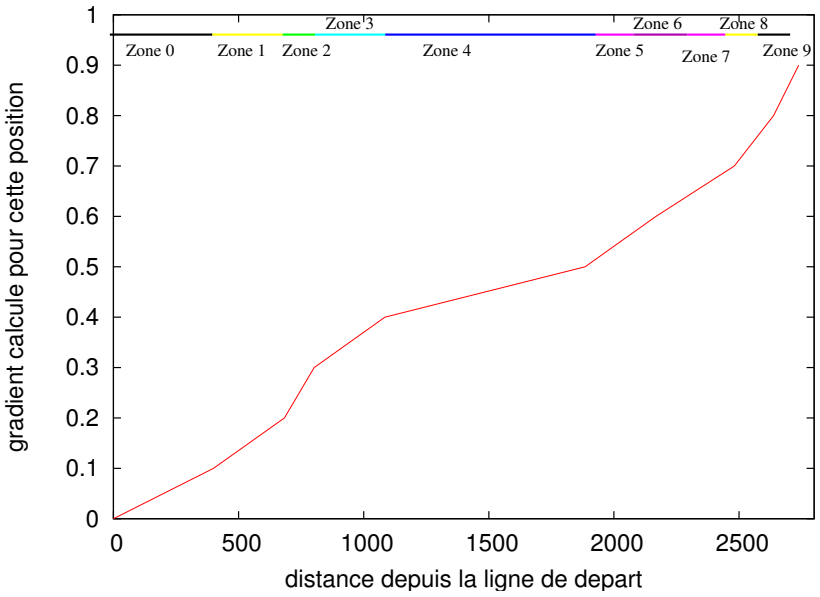


FIG. 5 – Quantité retournée par l'aide en fonction de la position de l'agent. Afin de simplifier la représentation de la fonction, on a supposé ici que le mobile se déplace uniquement à la corde du circuit.

stabilités intrinsèques au réseaux de neurones sont apparus. D'autres discrétisations, telles que la représentation curviligne, sont elles aussi en cours d'études. La figure 6, montre qu'en utilisant une stratégie d'exploration ϵ-glouton, l'architecture critique-critique parvient à stabiliser la longueur de la trajectoire en moins de 500 épisodes.
D'autre part, sur la figure 7, on voit que l'agent muni de l'architecture CC ne parvient pas à terminer un tour dès le premier essai. Cela est du au fait que notre agent CF n'a pas un contrôle total sur la vitesse.

4.4 Comparatifs avec les méthodes classiques

Afin de valider les résultats obtenus, nous comparons ces résultats avec ceux obtenus par un apprenant par renforcement seul. Pour cela, le circuit a été remodélisé comme un couloir droit et nous avons laissé un agent utilisant plusieurs algorithmes d'apprentissage par renforcement (Q(λ), PQ(λ)). Dans ces simulations, on ne tient pas compte des accélérations : les actions possibles sont simplement le déplacement d'une case dans une direction quelconque, du moment que le mobile ne quitte pas la piste. Le but est de voir si l'agent est capable de retrouver la sortie et de mesurer la longueur de la trajectoire obtenue par rapport à la trajectoire optimale. Les résultats sont présentés dans le tableau 8.

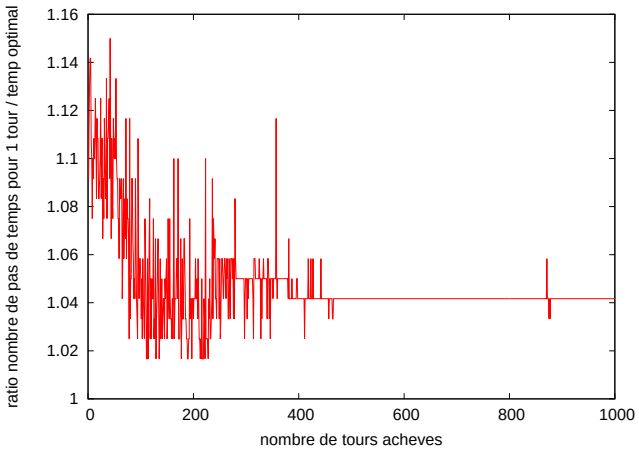


FIG. 6 – Nombre d'itérations pour atteindre la ligne d'arrivée par rapport à la solution optimale en fonction du nombre d'épisodes

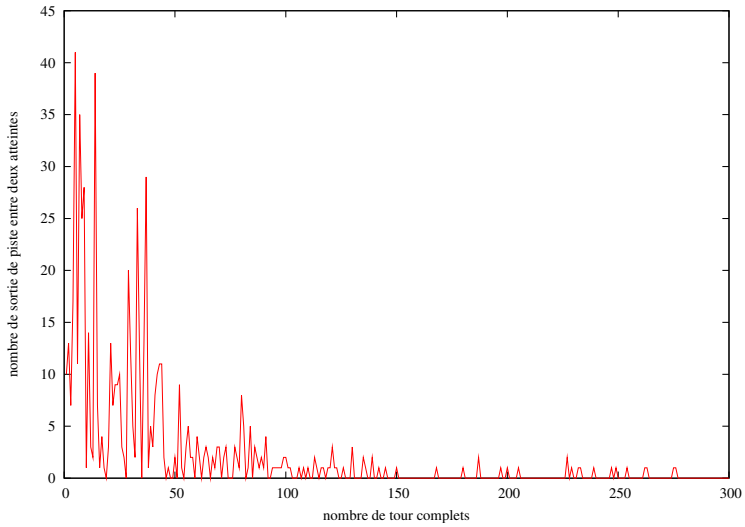


FIG. 7 – Nombre de sorties du circuit entre deux tours complets.

4.5 Résultats avec des aides incomplètes

Nous avons aussi mis à l'épreuve l'ACC en ne lui fournissant qu'une aide incomplète. Pour cela, nous avons redécoupé notre circuit en redivisant chacune des zones de 1 à 8 en 3 secteurs (exception faite des zones 2 et 9 qui, compte tenu de leurs faibles longueurs, n'ont été découpées qu'en deux secteurs) ce qui fait un total de 23 secteurs. Durant cette phase de test, certaines zones l'aide n'est pas disponible. Dans ce cas, CG transmet une valeur égale à celle du dernier état connu rencontré.

Le premier jeu de tests complémentaire consiste à supprimer un seul secteur, afin de déterminer

nb épisodes	Q(λ)	PQ(λ)	ACC
15	270	299.5	1.11
1000	34.8	30	1.05
10000	4.52	1.2	1.05
100000	1.53	1,12	1.05

FIG. 8 – Résultats de différents algorithmes par rapport à l'ACC : comparaison de la qualité des trajectoires obtenues

l'importance des renseignements dans les différentes zones du circuit. Les résultats présentés dans les figures 9 et 10 montrent que le passage le plus difficile est celui de la chicane : en effet à ce niveau les changements de directions sont nombreux ce qui oblige l'agent CF à une plus grande exploration de son environnement. Dans la chicane, l'aide est donc déterminante.

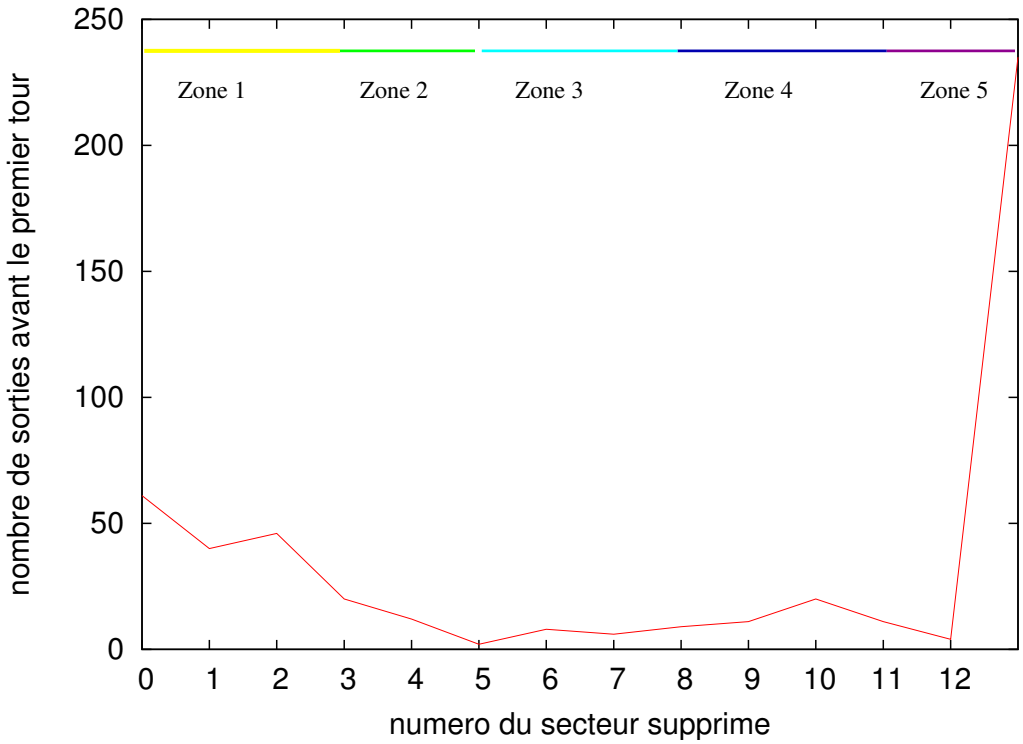


FIG. 9 – Nombre de sorties avant la première atteinte de la ligne d'arrivée avec une aide sur l'intégralité du circuit sauf secteurs indiqués en abscisses. Ici cela couvre les zones 1 à 5, c'est-à-dire avant la chicane

La suppression de deux secteurs ou plus non contigus n'influence que très peu les résultats. Ainsi, la suppréssion des secteurs 1 et 6 nous donne des résultats équivalents à ceux obtenus en supprimant uniquement la zone 1. Et même lorsqu'il s'agit de supprimer des zones entières, les résultats ne s'en

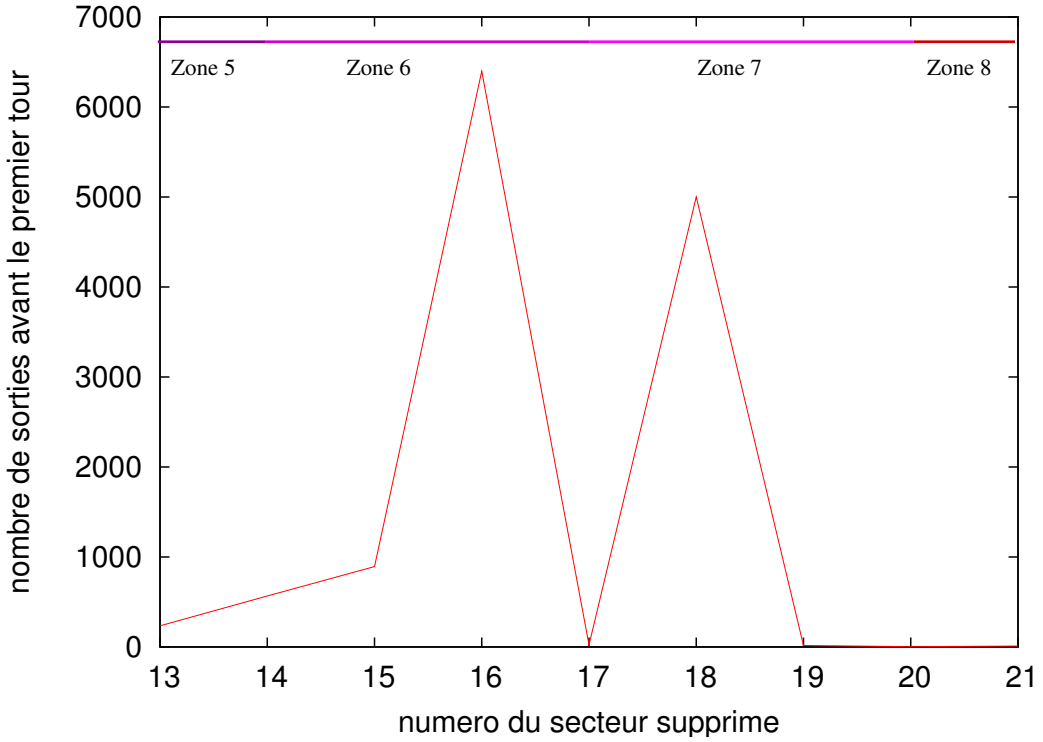
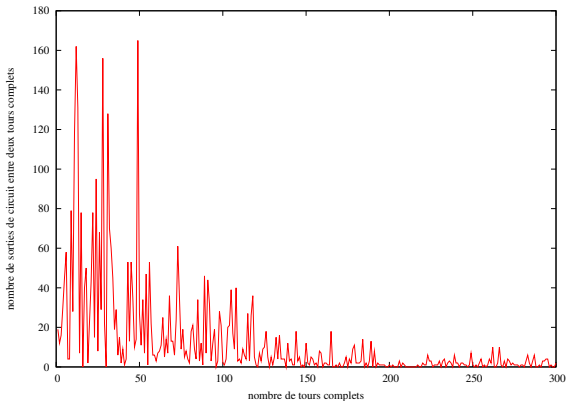


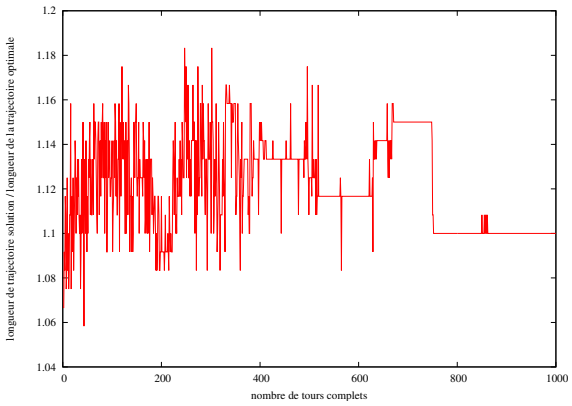
FIG. 10 – Nombre de sorties avant la première atteinte de la ligne d'arrivée avec une aide sur l'intégralité du circuit sauf secteurs indiqués en abscisses. Ici cela couvre les zones 6 à 8, c'est à dire depuis la chicane jusque la ligne d'arrivée

trouvent qu'amplifié. À titre d'exemples, sur la figure 11 sont présentés les résultats obtenus avec suppression de l'aide sur l'intégralité de la zone 4 (c'est-à-dire l'aide fournit la même valeur tout le long de la ligne droite).

Dans ce cas, même si la solution donnée pas l'agent est assez loin de la solution optimale ; on note qu'il ne lui faut que 19 échecs pour réaliser un premier tour complet. Malheureusement, l'utilisation d'un environnement tabulaire nous pose encore quelques problèmes : les solutions optimales ou proches de l'optimale, sont souvent proches d'une mauvaise solution d'où les instabilités qui apparaissent dans la figure 11.b et 6. D'autre part, lors de la suppression de zones complètes situées dans la chicane, l'agent ne parvient pas à trouver une politique pour résoudre cette tâche. Il semble que cela soit dû à la taille de l'espace de recherche à parcourir car alors que l'absence d'information dans la ligne droite ne lui pose pas de problème, dans la chicane il lui faut remettre en cause non seulement l'état courant mais aussi les états précédents sans trop perturber ce qu'il a déjà appris.



(a) nombre de sorties de route entre deux tours complets réussis



(b) nombre d'itérations nécessaires pour réussir un tour complet par rapport à la solution optimale en fonction du nombre d'épisodes

FIG. 11 – Résultats obtenus avec suppression de l'aide tout le long de la zone 4

5 Conclusion et discussion

Dans cette contribution, nous avons présenté une architecture, l'architecture critique-critique, basée sur l'apprentissage par renforcement et recevant de l'aide lui permettant d'atteindre plus rapidement une police de bonne qualité. Outre l'utiliastion d'une aide, une orginalité de cette architecture est son utilisation de plusieurs (ici deux) espaces d'états distincts pour représenter son apprentissage.

Au vue de nos expériences, il semble bien que l'architecture Critique-critique soit capable d'utiliser une aide pour assister un agent apprenant par renforcement à résoudre une tâche. D'autre part, l'utilisation de deux espaces d'états permet de définir assez simplement cette aide. Enfin, cette aide a permis de compenser la pauvreté de la fonction de retour de type « aiguille dans une meule de foin » : -1 en cas d'échec, 1 en cas réussite, 0 sinon (pour une tâche épisodique).

La combinaison des deux informations disponibles sur l'état courant doit être étudiée plus avant. À l'heure actuelle, nous avons considéré que l'apprenant par renforcement acquiert des corrections ou des confiances dans l'aide en chaque état, ce qui à amener l'utilisation d'addition ou de produit pour combiner ces deux informations.

L'utilisation de réseaux neuronaux [Tes92] ou d'autres approximateurs de fonctions devraient nous permettre de travailler plus aisément dans des espaces d'états (voire d'actions) continus. En attendant que nous puissions utiliser ces réseaux et afin d'accélérer encore l'apprentissage et de l'adapter à d'autres méthodes telles que le façonnage, l'algorithme PQ(λ) [Pre02] permettant la propagation des valeurs dans le TD(λ) est en cours d'implantation.

Dans la version actuelle, l'avis du premier critique est donné de façon explicite ce qui ne per-

met pas de retour d'apprentissage du second critique vers le premier. L'aide est donc utilisée de façon passive. Une fusion des deux phases d'apprentissage pourrait peut-être encore accélérer l'apprentissage. En tout cas, cela permettrait ue plus grande flexibilité dans la fourniture de l'aide à l'agent.

De plus, l'avis du premier critique est donné sous forme de valeurs numériques, ce qui nécessite une phase de préparation des données pour la construction d'un gradient par exemple. Concernant ce gradient, la question de savoir comment le construire pour qu'il soit le plus efficace (pour l'apprenant par renforcement) est à étudier ; pour l'instant, la solution utilisée est immédiate. D'autres formes de représentation de l'aide peuvent être envisagées. Liée au problème de la représentation est celui de la méthode la plus adéquate pour généraliser cette aide. S'agissant d'un problème de régression, on peut imaginer d'utiliser différentes techniques classiques, voire des méthodes de régression à vecteurs supports.

L'utilisation combinée de plusieurs critiques représentant l'aide est envisagée. On peut imaginer que chacun de ces critiques prennent en charge l'aide dans une région de l'espace d'états. On peut aussi imaginer de combiner l'avis de plusieurs critiques s'appuyant chacun sur l'aide qui a été fournie qu'il a généralisée selon son propre fonctionnement en utilisant des techniques de type *boosting* [Sch99].

Notons ici que l'utilisation de l'architecture critique-critique dans le contexte d'un POMDP ne pose pas de problème nouveau. On peut espèrer qu'elle améliore la résolution de ces problèmes ; une étude spécifique est nécessaire sur ce point.

Enfin, nous cherchons actuellement à déterminer si l'architecture critique-critique converge, et vers quoi, dans un problème de décision de markov.

Références

[BD99] A. Billard and K. Dautenhahn. Experiments in learning by imitation - grounding and use of communication in robotic agents. *Adaptive Behavior*, 7(3/4), 1999.

[BK96] P. Bakker and Y. Kuniyoshi. Robot see, robot do : An overview of robot imitation. In *AISB96 Workshop : Learning in Robots and Animals*, 1996.

[Clo97] J.A. Clouse. The role of training in reinforcement leanring. In J. Donahoe and V. Packard Dorsel, editors, *Neural-networks models of cognition*, pages 422–435. Elsevier, 1997.

[DC94] M. Dorigo and M. Colombetti. Robot shaping : developing autonomous agents through learning. *Artificial Intelligence*, 71 :321–370, 1994.

[Del00] Samuel Delepoulle. *Coopération entre agents adaptatifs ; étude de la sélection des comportements sociaux, expérimentations et simulations.* PhD thesis, Université de Lille 3, URECA, Villeneuve d'Ascq, October 2000. Thèse de doctorat de Psychologie.

[GMBQ97] P. Gaussier, S. Moga, J. Banquet, and M. Quoy. From perception-action loop to imitation processes : a bottom-up approach of learning by imitation. *Applied Artificial Intelligence*, 1(7), 1997.

[Gul97] V. Gullapalli. Reinforcement learning of complex behavior through shaping. In J. Donahoe and V. Packard Dorsel, editors, *Neural-networks models of cognition*, pages 302–314. Elsevier, 1997.

[HD01] L. Hugues and A. Drogoul. Shaping of robot behaviors by demonstrations. In *Proc. of the first Int'l Workshop on Epigenetic Robotics : Modeling cognitive development in robotic systems*, 2001.

[Lin92] L-J. Lin. Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning*, 8 :293–322, 1992.

[Mat94] M. Mataric. Reward functions for accelerated learning. In *Proc. of the* 11^{th} *Int'l Conf. on Machine Learning*, pages 181–189, 1994.

[MS96] R. Maclin and J.W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22 :251–282, 1996.

[PB03] B. Price and C. Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19 :569–629, 2003.

[Pom93] D.A. Pomerleau. *Knowledge-based training of artificial neural netowkrs for autonomous robot driving*. Kluwer Academic Press, 1993.

[Pre02] Ph. Preux. Propagation of q-values in tabular td(λ). In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proc.* 13^{th} *European Conference on Machine Learning (ECML)*, volume 2430 of *LNAI*, pages 369–380. Springer-Verlag, August 2002.

[RA98] J. Randlov and P. Alstrom. Learning to drive a bycicle using reinforcement learning and shaping. In *Proc. of the Int'l Conf. on Machine Learning*, 1998.

[RB02] M.T. Rosenstein and A.G. Barto. Supervised learning combined with an actor-critic architecture. Technical report, Dpt. of Computer Science, Univ. of Massachussets, Amherst, MA, USA, 2002.

[SB98] R. Sutton and A. Barto. *Reinforcement learning*. MIT Press, 1998.

[Sch97] Stefan Schaal. Learning from demonstration. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 1040. The MIT Press, 1997.

[Sch99] R.E. Schapire. A brief introduction to boosting. In *Proc. of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[SRT98] L.M. Saksida, S.M. Raymond, and D.S. Touretzky. Shaping robot behavior using principles from instrumental conditionning. *Robotics and autonomous systems*, 22(3/4), 1998.

[Tes92] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 8 :257–277, 1992.

[TM92] Sebastian B. Thrun and Knut Möller. Active exploration in dynamic environments. In John E. Moody, Steve J. Hanson, and Richard P. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 531–538. Morgan Kaufmann Publishers, Inc., 1992.