L'apprentissage par renforcement indirect dans les systèmes de classeurs

Olivier Sigaud et Pierre Gérard

AnimatLab (LIP6)
8, rue du Capitaine Scott
75015 PARIS

Abstract. Les systèmes de classeurs sont des systèmes à base de règles qui combinent une capacité d'apprentissage par renforcement et une capacité de généralisation. Au lieu d'associer des valeurs à des couples `(état, action)`, comme c'est le cas dans le cadre de l'apprentissage par renforcement tabulaire, ils associent des valeurs à des couples `(condition, action)`, dans lesquels la partie condition peut être vérifiée par plusieurs états, ce qui permet la généralisation. Les mécanismes d'apprentissage par renforcement habituels peuvent alors être appliqués à une telle représentation.
Récemment, une nouvelle famille de systèmes de classeurs, dits « à anticipation », est apparue, dans laquelle l'apprentissage par renforcement réalisé est un apprentissage indirect, ce qui signifie que l'agent construit un modèle de ses interactions avec son environnement, et se sert de ce modèle pour accélerer la convergence de l'apprentissage. Les architectures de la famille DYNA sont toutes construites sur ce modèle.
L'objet de cette contribution est de présenter un système de classeurs à anticipation, MACS, et de montrer comment on peut considérer MACS comme un cas particulier de la famille des architectures DYNA.

1 Introduction

Les problèmes d'apprentissage par renforcement mettent en jeu des agents adaptatifs impliqués dans une boucle sensori-motrice avec leur environnement (voir figure 1) [Kaelbing et al., 1996,Sutton & Barto, 1998]. La représentation formelle de tels problèmes est donnée par le cadre des processus de décision markoviens, qui définissent les éléments suivants :

– un espace fini d'états S;
– un ensemble fini d'actions A;
– une fonction de transition $t : S \times A \rightarrow \Pi(S)$ où $\Pi(S)$ est une distribution de probabilités sur S;
– une fonction de récompense $r : S \times A \times S \rightarrow \mathbb{R}$ qui associe une récompense immédiate à chaque transition.

Parmi les algorithmes d'apprentissage par renforcement, l'un des plus populaires est *Q-learning* [Watkins, 1989]. Cet algorithme met à jour incrémentalement une « *Q-table* » représentant une fonction de qualité $q : S \times A \rightarrow \mathbb{R}$, sans élaborer de modèle ni des transitions, ni des récompenses. La qualité $Q(s, a)$ représente la récompense cumulée qu'un agent peut espérer recevoir quand il réalise l'action a dans l'état s, et suit par la suite une politique optimale. La valeur d'un état s est le maximum sur a des qualités $Q(s, a)$ associées à cet état.

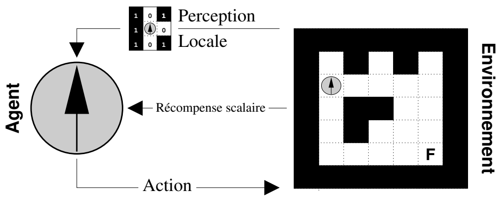

Fig. 1. Dans les problèmes d'apprentissage par renforcement, un agent est situé dans un environnement qu'il ne connaît pas. A chaque pas de temps, il perçoit une situation et réalise une action avant le pas de temps suivant. Les buts de l'agent sont définis par des récompenses scalaires fournies par l'environnement. Son objectif global est d'apprendre un comportement qui lui permet de maximiser ses récompenses. Ici, l'environnement est constitué de cases et la récompense est une source de nourriture (notée F), mais l'apprentissage par renforcement s'applique plus généralement à la recherche de politiques optimales dans des graphes quelconques.

Lorsqu'on utilise un système de classeurs pour résoudre un problème de décision markovien, l'organe de décision de l'agent est constitué d'un ensemble de règles appelées « classeurs ». Ces classeurs sont composés au moins d'une partie `[Condition]` et d'une partie `[Action]`. La partie `[Condition]` contient plusieurs attributs. En conséquence de cette représentation, l'agent n'a pas directement accès à l'état du processus, mais à une *situation* composée des attributs qu'il perçoit (voir figure 2).

Vis-à-vis des algorithmes d'apprentissage par renforcement tabulaires comme *Q-learning*, le principal avantage des systèmes de classeurs provient de leur capacité de généralisation. Dans la partie `[Condition]` d'un classeur, chaque symbole spécifie la valeur que doit avoir l'attribut considéré pour que le classeur puisse être choisi. Mais il existe un symbole particulier, noté `#`, qui indique que l'attribut peut être ignoré dans le processus d'appariement du classeur avec la situation observée. Par conséquent, un classeur dont la partie `[Condition]` contient un ou plusieurs `#` peut être choisi dans plusieurs situations distinctes. On dit d'un classeur dont la partie `[Condition]` contient des `#` qu'il est plus général qu'un classeur qui en est dépourvu. Par exemple, un classeur dont la partie `[Condition]` est `[#1#0]` peut être apparié avec quatre situations alors qu'un classeur dont la partie `[Condition]` est `[#110]` ne peut être apparié qu'avec deux situations.

Le processus de généralisation consiste à déterminer où peuvent être positionnés les `#` sans détériorer la performance de l'agent. Grâce à ce processus, il est possible de construire un modèle de la récompense attendue plus compact que la *Q-table* de l'apprentissage par renforcement tabulaire, chaque classeur `[Condition][Action] q` modélisant plusieurs triplets `(état, action, valeur)` de la *Q-table*.

En règle générale, la qualité associée à chaque classeur est apprise grâce à différents algorithmes de propagation de la récompense, comme *Bucket Brigade* [Holland, 1985] ou *Q-learning*, dans XCS [Wilson, 1995] par exemple.

Mais les algorithmes de type *Q-learning* propagent très lentement la qualité des différentes situations. La qualité d'un couple `(état, action)` est mise à jour uniquement lorsqu'un agent réalise effectivement cette action à partir de cet état, et la mise à jour s'effectue uniquement en fonction de la qualité de l'état d'arrivée. Un système

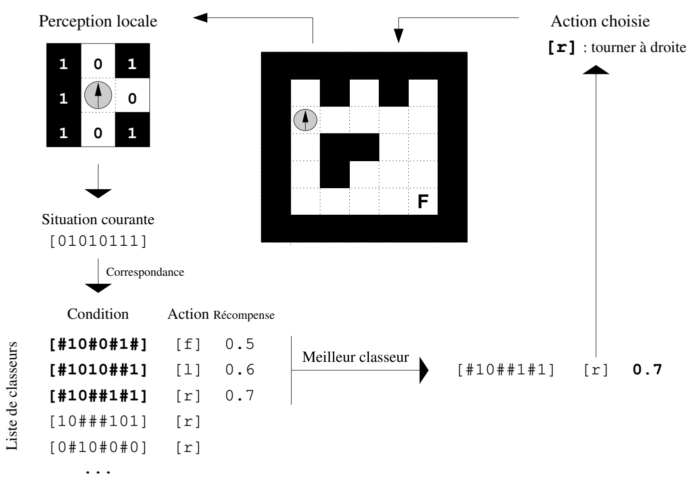

Fig. 2. Un système de classeurs contient une liste de classeurs. L'agent perçoit une situation sous la forme d'un ensemble d'attributs. Dans cet exemple, l'agent est situé dans un labyrinthe et perçoit la présence (symbole 1) ou l'absence (symbole 0) de blocs dans les huit cases environnantes. Les cases sont considérées dans le sens des aiguilles d'une montre, en commençant par le nord, si bien qu'il perçoit [01010111] dans l'exemple ci-dessus. L'agent doit choisir à chaque instant entre avancer ([f]) tourner à droite ([r]) ou tourner à gauche ([l]). Parmi les classeurs appris, le système identifie ceux dont les conditions sont compatibles avec la situation courante, puis choisit dans cet ensemble un classeur dont la prédiction de récompense est élevée. L'action spécifiée par ce classeur est activée si bien que, dans l'exemple, l'agent se tourne vers sa droite.

capable de mettre à jour plusieurs qualités à chaque pas de temps permettrait un apprentissage beaucoup plus rapide. La capacité d'apprentissage de la plupart des systèmes de classeurs repose sur le *Q-learning*. Par conséquent, ils souffrent du même problème de lenteur de convergence que cet algorithme.

L'objet de cet article est de montrer comment l'apprentissage par renforcement indirect, qui permet de résoudre ce problème de lenteur, se trouve réalisé dans le cadre des systèmes de classeurs par une famille de systèmes de classeurs dits « à anticipation ». Dans la section suivante, nous présentons l'apprentissage par renforcement indirect et sa source psychologique, l'apprentissage latent. Puis nous présentons à la section 3 le système de classeurs MACS, qui est capable d'apprentissage par renforcement indirect. Nous donnons quelques résultats expérimentaux représentatifs et nous indiquons des directions de recherche future.

2 Apprentissage par renforcement indirect

2.1 L'apprentissage latent

Nous avons dit en introduction que des algorithmes d'apprentissage par renforcement tels que *Q-learning* sont lents car ils requièrent d'un agent qu'il expérimente effectivement un grand nombre de transitions pour rétropropager les récompenses immédiates et calculer la fonction qualité qui en découle.

Plutôt que d'attendre qu'un agent effectue des transitions, une façon d'accélérer la propagation de la valeur des situations consiste à construire un modèle des transitions réalisées par l'agent et à se servir de ce modèle pour appliquer un algorithme de propagation indépendamment du comportement de l'agent. En effet, quand un agent interagit avec son environnement, ses actions ne débouchent pas seulement sur une possible punition ou récompense, mais aussi sur une situation ultérieure. L'agent peut donc construire un modèle des transitions dont il fait l'expérience, indépendamment de tout modèle des récompenses.

Cette idée de construire un modèle trouve sa source dans des travaux de psychologie. En opposition marquée avec les théories béhavioristes qui dominaient à son époque, Tolman [Tolman, 1932] présentait l'apprentissage comme un processus d'acquisition d'un modèle interne des relations entre situations et actions. Là où le béhaviorisme supposait le renforcement du déclenchement d'une réponse à partir d'un stimulus donné, donc le renforcement d'une association Stimulus-Réponse, Tolman considérait plutôt la mémorisation de relations de type Stimulus-Réponse-Stimulus, indiquant que telle réponse déclenchée dans telle situation débouche sur telle nouvelle situation. Ce processus de mémorisation mis en évidence empiriquement par Seward [Seward, 1949] est appelé « apprentissage latent », car on montre qu'il opère même en l'absence de punition ou de récompense.

2.2 Systèmes de classeurs à anticipation

Le premier modèle d'apprentissage latent réalisé avec des systèmes de classeurs est le système CFSC2, de Riolo [Riolo, 1991], directement inspiré du cadre original des systèmes de classeurs à messages internes proposés par Holland [Holland, 1990].

Plus récemment, plusieurs systèmes de classeurs capables de réaliser un apprentissage latent sont apparus, à commencer par ACS [Stolzmann, 1998,Butz et al., 2000] qui a évolué pour devenir ACS2 [Butz, 2002]. Au cœur d'ACS, l'algorithme ALP (*Anticipatory Learning Process*) est une formalisation de la théorie psychologique du contrôle prédictif du comportement développée par Hoffmann [Hoffmann, 1993].

Dans ACS, les classeurs disposent d'une partie `[Condition]`, une partie `[Action]` et une partie `[Effet]`. La partie `[Effet]` représente les effets attendus de l'action dans toutes les situations qui vérifient la partie `[Condition]` du classeur. Ce formalisme est proche des règles `[contexte-action-résultat]` de Drescher [Drescher, 1991], mais s'y ajoute la capacité de généralisation spécifique des systèmes de classeurs.

Nos systèmes de classeurs, YACS [Gérard & Sigaud, 2001,Gérard et al., 2002] et MACS [Gérard et al., 2003] sont proches d'ACS et ACS2. Leurs classeurs disposent aussi des parties `[Condition]`, `[Action]` et `[Effet]`. La principale originalité de ces deux derniers systèmes consiste en une séparation plus claire entre deux processus d'apprentissage bien distincts : l'apprentissage d'un modèle des transitions et l'apprentissage d'un modèle des récompenses.

Nous avons dit que la principale originalité des systèmes de classeurs réside dans leur capacité de généralisation. Dans les systèmes de classeurs comme XCS, la généralisation consiste à regrouper sous une même partie `[Condition]` les situations pour lesquelles l'espérance de récompense est similaire. Dans un système de classeurs à anticipation, au contraire, il est possible de chercher à généraliser le modèle des transitions, en regroupant sous une même partie `[Condition]` les situations qui conduisent au même effet. Le constat qui a donné naissance à YACS et MACS est que ces deux processus de généralisation sont indépendants l'un de l'autre et gagnent à être effectués séparément. Il en résulte la possibilité de séparer complètement l'apprentissage du modèle des transitions et celui du modèle des récompenses[1].

Cette clarification se traduit concrètement par des performances sensiblement meilleures sur une série de benchmarks, puisque YACS a besoin de réaliser trois fois moins d'expériences qu'ACS pour apprendre un modèle complet des transitions qui s'avère en outre sensiblement plus compact.

2.3 Les architectures Dyna

L'idée d'une séparation claire entre l'apprentissage d'un modèle des transitions et l'apprentissage d'un modèle des récompenses était déjà connue dans le cadre de l'apprentissage par renforcement tabulaire, avec les architectures Dyna [Sutton, 1991]. En effet, Sutton a proposé cette famille d'architectures pour doter un agent de la capacité à mettre à jour plusieurs qualités en même temps, de façon à accélérer sensiblement l'apprentissage de la fonction qualité.

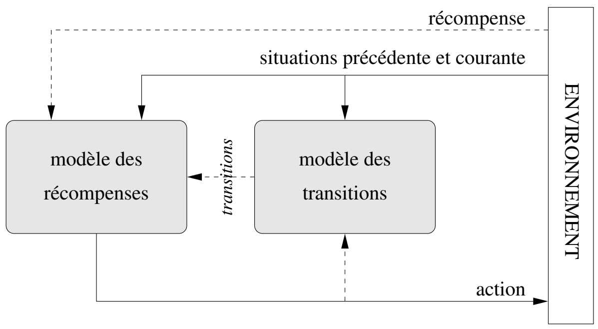

Fig. 3. L'architecture Dyna combine un modèle des récompenses et un modèle des transitions. Le modèle des transitions est utilisé pour accélérer le processus d'apprentissage par renforcement. La construction des deux modèles requiert une mémorisation de la situation précédente, ce qui n'est pas indiqué sur la figure.

L'architecture Dyna, présentée sur la figure 3, construit un modèle des transitions effectuées par l'agent dans son environnement et un modèle des récompenses qu'il peut y obtenir.

Le modèle des transitions prend la forme d'une liste de triplets (s_t, a_t, s_{t+1}) qui indiquent que, si l'agent effectue l'action a_t dans l'état s_t, il atteindra immédiatement l'état s_{t+1}. Dans le contexte des systèmes de classeurs à anticipation, le même modèle est représenté par une liste de classeurs séparés en parties `[Condition]`, `[Action]` et `[Effet]`. Par conséquent, la principale différence entre les modèles des transitions dans une architecture Dyna et dans un système de classeurs à anticipation provient

[1] Plus précisément, dans YACS et MACS nous avons cherché à atteindre le degré maximal de généralisation dans le modèle des transitions, sans nous soucier de généraliser le modèle des récompenses. Récemment, Butz [Butz, 2003] a proposé un modèle qui utilise conjointement les deux formes de généralisation.

de la capacité de généralisation des classeurs, qui permet de construire un modèle des transitions plus compact.

Une fois appris, le modèle des transitions peut être utilisé indépendamment de la situation courante de l'agent pour simuler des actions et ainsi propager les valeurs de différentes situations à partir de diverses sources de récompense. En pratique, ce processus de propagation consiste à réaliser des actions simulées à partir de situations fictives, éventuellement plusieurs fois par pas de temps. Le modèle des récompenses consiste en une liste de toutes les situations possibles auxquelles sont associés la récompense immédiate et la récompense attendue que l'agent peut obtenir à partir de ces situations.

A partir du point de vue qui vient d'être développé, les systèmes de classeurs à anticipation peuvent être vus comme combinant deux propriétés cruciales pour les systèmes d'apprentissage par renforcement. Comme les architectures Dyna, ils apprennent un modèle des transitions, ce qui les dote de propriétés d'anticipation et leur permet d'apprendre plus vite. Comme les systèmes de classeurs classiques, ils sont dotés d'une capacité de généralisation, ce qui leur permet de construire un modèle plus compact que les architectures Dyna tabulaires.

3 Présentation succincte de MACS

3.1 Formalisme de MACS

Dans ACS, ACS2 et YACS, la partie `[Effet]` de chaque classeur est organisée de façon à indiquer quels sont les attributs qui changent et quels sont ceux qui ne changent pas si l'on applique l'action dans une situation compatible avec la partie `[Condition]` du classeur. A cette fin, les parties `[Effet]` peuvent contenir un symbole "=", qui signifie que l'attribut ne change pas [Gérard, 2001]. Par exemple, le classeur `[#0#1] [0] [=10=]` appliqué à la situation 1001 prédit que la situation résultant de son application sera 1101.

L'originalité la plus manifeste de MACS provient de la transformation opérée sur la sémantique de la partie `[Effet]` des classeurs. Alors que les systèmes précédemment cités ne peuvent exprimer une relation qu'entre la valeur d'un attribut et la valeur du même attribut au pas de temps suivant, il existe des régularités qui portent sur des relations entre les valeurs de plusieurs attributs différents lors de deux pas de temps successifs.

Par exemple, si l'agent représenté sur la figure 2 se tourne sur la droite, le mur qu'il voyait à sa gauche est à présent derrière lui.

Dans MACS, on représente une régularité de ce type avec un classeur tel que `[####1###]` [$\curvearrowright$] `[??1?????]`, où le symbole "?" signifie que le classeur ne sait pas prédire la valeur de l'attribut considéré. L'introduction de ce nouveau symbole permet donc de prédire séparément la valeur des différents attributs au pas de temps précédent.

Dans MACS toujours, nous avons fait le choix de prédire la valeur de chaque attribut indépendamment de celle des autres. Comme cela apparaît sur la figure 4, cela permet de décomposer le modèle des transitions complet en un ensemble de modèles dont chacun réalise une fonction qui prédit la valeur d'un attribut particulier. L'avantage de ce nouveau formalisme est qu'il permet d'exprimer un nombre de régularités beaucoup plus grand que le précédent, si bien que le modèle des transitions que l'on obtient est plus compact et s'apprend plus vite.

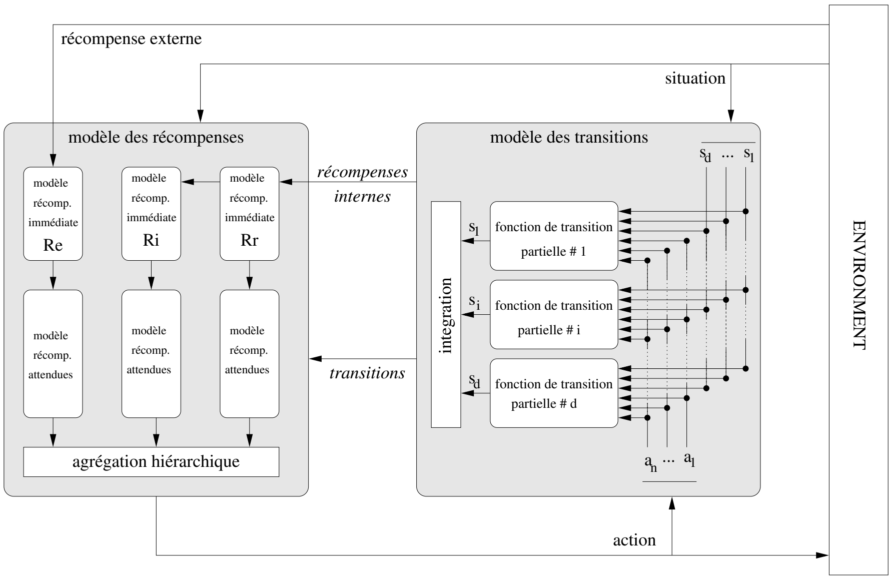

Fig. 4. L'architecture globale de MACS.

3.2 Apprentissage du modèle des transitions

Le processus d'apprentissage du modèle des transitions réalisé dans MACS s'apparente à celui qui est réalisé dans Dyna, à ceci près que le modèle appris est constitué de classeurs aussi généraux que possibles. Plus précisément, le modèle des transitions doit être à la fois aussi général que possible et aussi exempt d'erreur que possible. Pour obtenir un tel modèle, plutôt que de faire appel à un algorithme génétique, comme le font la plupart des systèmes de classeurs, nous nous efforçons d'obtenir le bon degré de généralité grâce à la combinaison de deux heuristiques :

- on applique aux classeurs qui ne sont pas assez précis une heuristique de spécialisation des conditions ;
- on applique aux classeurs qui ne sont pas assez généraux une heuristique de généralisation des conditions.

L'utilisation conjointe de ces deux heuristiques permet de faire converger la généralité et la précision des classeurs à un niveau optimal.

Chacune des heuristiques repose sur une mémoire partielle des dernières prédictions correctes et incorrectes de chaque classeur, ainsi que sur des estimations destinées à déterminer si un classeur donné doit être généralisé davantage ou spécialisé davantage et, le cas échéant, sur quel attribut doit porter la modification. Pour plus de détails sur les mécanismes qui permettent la réalisation de ces deux heuristiques, voir [Gérard, 2002].

3.3 Compromis exploration-exploitation

Comme c'est le cas dans Dyna, le modèle des transitions est utilisé dans MACS pour accélerer l'apprentissage du modèle des récompenses. Afin de fonctionner efficacement sur des problèmes de grande taille, MACS est doté d'un mécanisme d'exploration active qui repose sur une politique combinant hiérarchiquement trois critères :

- L'agent choisit en priorité les actions qui lui apportent de l'information sur des transitions qu'il n'a pas encore effectuées assez souvent. Cela lui permet de constituer son modèle des transitions le plus vite possible.
- Il choisit ensuite les actions qui lui permettent de maximiser sa récompense externe sur le long terme, ce qui correspond à l'objectif de tout apprentissage par renforcement.
- Enfin, si plusieurs actions s'avèrent équivalentes vis-à-vis des deux critères précédents, il choisit de préférence les actions qu'il n'a pas choisies depuis le plus longtemps, afin de mettre à jour son modèle des transitions lorsque celui-ci évolue.

A chacun de ces critères est associée une fonction de récompense immédiate et un processus d'itération de la valeur qui permet d'en déduire une récompense attendue sur le long terme. Là encore, les détails du calcul des critères et des mécanismes sont fournis dans [Gérard, 2002].

4 Résultats expérimentaux

Parmi les nombreux résultats expérimentaux obtenus avec MACS, nous présentons dans cette section des résultats d'expériences dans lesquelles on déplace des sources de récompense, afin d'illustrer l'apport conjoint du mécanisme d'exploration active et du mécanisme d'apprentissage latent responsable de la construction d'un modèle des transitions.

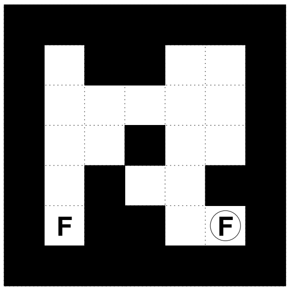

Fig. 5. Maze216C

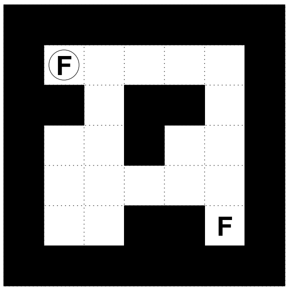

Fig. 6. Maze228C

Nous avons testé MACS sur les deux problèmes jouets apparaissant sur les figures 5 et 6. De tels environnements sont couramment utilisés pour valider le fonctionnement de systèmes de classeurs. Au début de l'expérience, la source de récompense est située dans la case marquée d'un F entouré d'un cercle. Dès que l'agent l'a atteint vingt fois, elle est déplacée dans la case marquée d'un F simple. On la redéplace ainsi d'un emplacement à l'autre à chaque fois que l'agent la rejoint vingt fois.

La figure 7 présente le nombre de pas de temps dont MACS a besoin pour atteindre la récompense dans Maze216C et Maze228C sur 250 essais successifs. Les résultats sont des moyennes sur 100 expériences, ce qui explique les légères variations de performance.

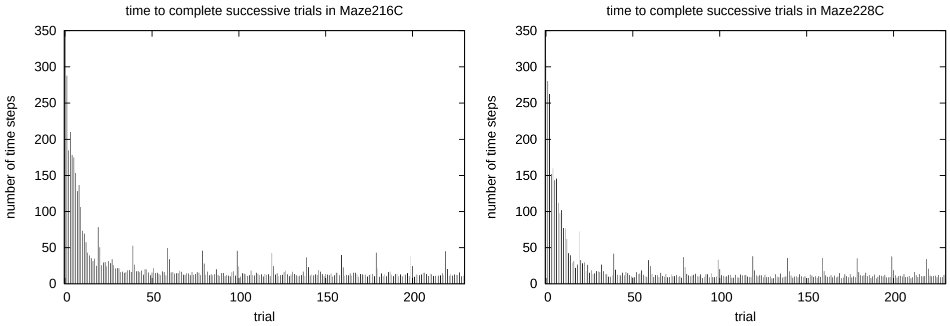

Fig. 7. Nombre de pas de temps nécessaires à MACS pour rejoindre la récompense lors d'essais successifs sur Maze216C et Maze228C.

Les résultats montrent que, bien qu'il ne réalise qu'un pas de *Value Iteration* par pas de temps, MACS est capable de retrouver la source de nourriture très rapidement.

5 Conclusion

L'objectif principal de cet article était de montrer comment un système de classeurs à anticipation peut être vu comme un cas particulier d'architecture DYNA doté d'une capacité supplémentaire de généralisation.

Dans ce contexte, MACS apparaît comme un système de classeurs à anticipation particulièrement efficace grâce à plusieurs caractères distinctifs.

- Son formalisme de représentation des effets lui permet de construire un modèle des transitions plus compact que ACS et YACS.
- Les heuristiques qu'il utilise lui permettent de construire ce modèle beaucoup plus rapidement que ses concurrents.
- La combinaison hiérarchique de fonctions de récompense lui permet de résoudre très efficacement le problème du compromis exploration-exploitation.

Au delà de ces qualités spécifiques, l'architecture de MACS présentée sur la figure 4 a permis de dégager une avancée conceptuelle plus générale. Il s'avère que chacun des modules représentés sur la figure 4 peut être réalisé à l'aide d'un mécanisme d'approximation de fonction. Nos travaux futurs consisteront donc à réaliser un nouveau système dont l'architecture correspond à celle présentée sur la figure 4, mais dont les modules sont réalisés par un unique composant d'approximation de fonction générique, faisant appel à des méthodes d'estimation statistique incrémentale.

References

[Butz, 2002] Butz, M. V. (2002). Biasing Exploration in an Anticipatory Learning Classifier System. Edité dans Lanzi, P. L., Stolzmann, W., & Wilson, S., editeurs, *Advances in Learning Classifier Systems*, volume 2321 of *Lecture Notes in Artificial Intelligence*, pages 3–22. Springer-Verlag, Berlin.

[Butz, 2003] Butz, M. V. (to appear, 2003). Generalized State Values in an Anticipatory Learning Classifier System. Edité dans Butz, M. V., Sigaud, O., & Gérard, P., editeurs, *LNAI 2684: Anticipatory Behavior in Adaptive Learning Systems*. Springer-Verlag.

[Butz et al., 2000] Butz, M. V., Goldberg, D. E., & Stolzmann, W. (2000). Introducing a genetic generalization pressure to the Anticipatory Classifier System part I: Theoretical approach. Edité dans *Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 34–41.

[Drescher, 1991] Drescher, G. L. (1991). *Made-Up Minds: A constructivist approach to Artificial Intelligence*. MIT Press, Cambridge MA.

[Gérard, 2001] Gérard, P. (2001). Generalisation and Latent Learning in Learning Classifier Systems. Edité dans *Proceedings of the XIX EURO Summer Institute*, Toulouse, France.

[Gérard, 2002] Gérard, P. (2002). *Apprentissage latent et apprentissage par renforcement dans les systèmes de classeurs*. Thèse de troisième cycle de l'Université PARIS VI.

[Gérard et al., 2003] Gérard, P., Meyer, J.-A., & Sigaud, O. (2003). Combining latent learning with dynamic programming. *European Journal of Operation Research*, to appear.

[Gérard & Sigaud, 2001] Gérard, P. & Sigaud, O. (2001). YACS : Combining Anticipation and Dynamic Programming in Classifier Systems. Edité dans Lanzi, P. L., Stolzmann, W., & Wilson, S., editeurs, *Advances in Learning Classifier Systems*, volume 1996 of *Lecture Notes in Artificial Intelligence*, pages 52–69. Springer-Verlag, Berlin.

[Gérard et al., 2002] Gérard, P., Stolzmann, W., & Sigaud, O. (2002). YACS: a new Learning Classifier System with Anticipation. *Journal of Soft Computing : Special Issue on Learning Classifier Systems*, 6(3-4):216–228.

[Hoffmann, 1993] Hoffmann, J. (1993). *Vorhersage und Erkenntnis [Anticipation and Cognition]*. Hogrefe.

[Holland, 1985] Holland, J. H. (1985). Properties of the bucket brigade algorithm. Edité dans Grefenstette, J. J., editeur, *Proceedings of the 1st international Conference on Genetic Algorithms and their applications (ICGA85)*, pages 1–7. L.E. Associates.

[Holland, 1990] Holland, J. H. (1990). Concerning the emergence of tag mediated lookahead in Classifier Systems. *Special Issue of Physica D*, 42:188–201.

[Kaelbing et al., 1996] Kaelbing, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement Learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

[Riolo, 1991] Riolo, R. L. (1991). Lookahead planning and latent learning in a Classifier System. Edité dans Meyer, J.-A. & Wilson, S. W., editeurs, *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptative Behavior*, pages 316–326. MIT Press.

[Seward, 1949] Seward, J. P. (1949). An experimental analysis of latent learning. *Journal of Experimental Psychology*.

[Stolzmann, 1998] Stolzmann, W. (1998). Anticipatory Classifier Systems. Edité dans Koza, J., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., & Riolo, R., editeurs, *Genetic Programming*, pages 658–664. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[Sutton, 1991] Sutton, R. S. (1991). Reinforcement learning architectures for animats. Edité dans Meyer, J.-A. & Wilson, S. W., editeurs, *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptative Behavior*, pages 288–296, Cambridge, MA. MIT Press.

[Sutton & Barto, 1998] Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[Tolman, 1932] Tolman, E. C. (1932). *Purposive behavior in animals and men*. Appletown, New York.

[Watkins, 1989] Watkins, C. J. (1989). *Learning with delayed rewards*. Thèse de doctorat, Psychology Department, University of Cambridge, England.

[Wilson, 1995] Wilson, S. W. (1995). Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175.