Apprendre à s'adapter dans les Modèles de Markov

Alain Dutech
dutech@loria.fr
LORIA, équipe Maia

25 février 2002

Mots clef : adaptation, apprentissage par renforcement, Processus Décisionnel de Markov.

1 Introduction

Notre recherche est centrée sur la notion d'agent. Un agent est une entité artificielle, souvent un programme, qui est amenée à évoluer dans un environnement pour y accomplir différentes tâches. L'agent est au cœur d'une boucle sensori-motrice et doit essayer d'agir avec une certaine rationalité dans toutes les situations.

C'est pourquoi la notion d'adaptation à l'environnement est cruciale pour un agent. En effet, un environnement peut être aléatoire, mal connu ou dynamique, autant de problèmes à résoudre pour que l'agent reste opérationnel. Parmi les solutions possibles pour prendre en compte ces divers inconvénients nous avons choisi un formalisme stochastique s'appuyant sur les Processus Décisionnels de Markov (MDP).

Dans un premier temps (Section 2) nous parlerons des différents aspects de l'environnement que nous essayons de prendre en compte. Ensuite, nous présenterons les MDP et l'apprentissage par renforcement (Section 4) avant de terminer par des propositions pour essayer de l'adapter à des environnements dynamiques.

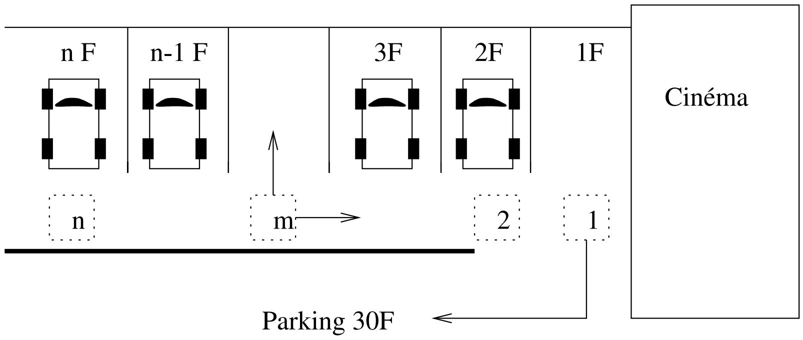
2 Un environnement hostile

Pour un agent donné, l'environnement est constitué de tout ce avec quoi (ou qui) il peut interagir. Bien souvent, cet environnement n'est pas bienveillant pour l'agent, au sens où il ne lui facilite pas la tâche. Au contraire, il est une source constante de problèmes à résoudre.

D'une part, cet environnement, même parfaitement connu, est bien souvent aléatoire. L'aléa trouve son origine dans l'environnement lui-même (passage inopiné de jambes humaines devant la caméra d'un robot) et dans les interactions entre l'agent et l'environnement. Ainsi, les capteurs dont sont équipés les agents sont rarement parfaits et les actions prévues par les agents trouvent facilement une issue non prévue (glissement des roues, agent poussé par un autre agent...). Les Processus Décisionnels de Markov, décrits à la Section 3, sont une solution pour modéliser ce genre d'environnement aléatoire afin d'y rechercher un comportement optimal pour notre agent.

De plus, l'environnement est souvent difficile à modéliser. La raison majeure est la complexité de la réalité que l'on veut modéliser. Il peut aussi s'agir d'un environnement que l'on ne connaît pas et que l'on doit explorer. Dans tous ces cas, on ne peut plus employer les algorithmes classiques des MDP et il faut alors se tourner vers d'autres techniques. Nous présentons ici des techniques d'apprentissage par renforcement, utilisées car elles s'appuient aussi sur le formalisme des MDP, comme nous le détaillons en Section 4.

Par essence, l'environnement est dynamique. Cette dynamicité, qui s'exprime par une modification permanente du modèle utilisé, n'est que difficilement prise en compte par les algorithmes précédents. Elle est pourtant courante dans les problèmes que l'on rencontre, et surtout dans une classe de problèmes



Etats : [m, occ|lib|gar]

Actions : Avancer, Garer

Récompense : xF

Transitions :

T([n,.], A, [n-1, lib]) = 0.3

T([n,.], A, [n-1, occ]) = 0.7

T([n,lib], G, [n,gar]) = 1.0

FIG. 1 – **Exemple de MDP.** *On veut se garer le plus près possible du cinéma pour payer le moins cher de parking. A chaque emplacement on peut se garer ou avancer, mais on ne sait pas à l'avance si la place suivante sera libre ou occupée. Les transitions prennent en compte l'aspect aléatoire de l'environnement : une place a 70% de chance d'être occupée et 30% d'être libre. Si on n'a pas de place on doit aller au Parking. La question est : quand faut-il commencer à se garer si on trouve une place libre?*

qui nous tient à cœur : les systèmes multi-agents [Ferber, 1995]. Les autres agents sont souvent la source des modifications de l'environnement (déplacement d'obstacles, fermeture ou ouverture de portes...). La Section 6 essaie d'apporter des éléments de réponse à ce problème des plus importants et qui est un thème de recherche à peine exploré.

3 Processus Décisionnels de Markov

Un Processus décisionnel de Markov [Puterman, 1994] est composé de plusieurs ensembles finis :

- $\mathcal{S}$ est l'ensemble des états du système
- $\mathcal{A}$ est l'ensemble des actions possibles pour l'agent

Les changements d'états du systèmes sont gouvernés par les actions qui déterminent les probabilités de transition entre les états selon la formule suivante: $P\{S_{t+1} = s'|A_t = a, S_t = s\} = \mathcal{T}(s,a,s')$ où $\mathcal{T}$ est une fonction de $\mathcal{S} \times \mathcal{A} \times \mathcal{S}$ dans [0,1]. C'est ainsi que l'on tient compte de l'aspect aléatoire de l'environnement. L'agent reçoit aussi une récompense r , qui est une fonction réelle, après chaque action.

Le but pour l'agent est de choisir, dans chaque état, l'action qui permet d'optimiser (maximiser ou minimiser) une certaine fonction V de la récompense qu'on appelle l'utilité. Un exemple classique, qui permet de tenir compte des récompenses futures est $V(s) = \sum_{t=0}^{\infty} \gamma^t r(S_t|S_0 = s)$ avec γ un réel de [0,1[. On cherche ainsi une politique π de $\mathcal{S}$ dans $\mathcal{A}$. La figure 1 donne un exemple de MDP pour un problème simple.

De nombreux algorithmes issus de la programmation dynamique permettent de calculer l'utilité optimale pour un MDP donné en utilisant le fait que cette utilité optimale V^* est solution de l'équation de Bellman : $V^*(s) = opt_a\{r(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')V^*(s')\}$. Les algorithmes les plus connus sont *value iteration* et *policy iteration*. Une fois l'utilité optimale connue, on peut aisément en déduire la politique optimale par : $\pi(s) = argopt_a\{r(s) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,a,s')V^*(s')\}$.

La figure 2 donne un exemple de calcul *à la main* de l'utilité pour l'exemple précédent, ainsi que la politique optimale qui en découle.

Ce formalisme a été appliqué avec succès à plusieurs problèmes, notamment dans les domaines de la robotique mobile, de l'allocation de ressources, de routages de paquets dans un réseau, de la gestion d'une flotille d'ascenseurs... Cependant, ces algorithmes souffrent d'une grande limitation : les fonctions de transition $\mathcal{T}$ et de récompense r doivent être connues. Or ce n'est pas toujours le cas. L'apprentissage par renforcement permet de dépasser cette limite.

Si on n'a pas de place, il faut aller au Parking:
V*([0]) = 30
Pour la place no 1:
    Elle est prise:
    V*([1,occ]) = min(T([1,occ], A, [0]) V*([0])) = V*(0) = 30
    On est garé:
    V*([1,gar]) = 1
    Elle est libre, on a le choix:
    V*([1,lib]) = min{ T([1, lib], A, [0]) V*([0]), T([1,lib], G, [1,gar]) V*([1,gar]) } = 1
De manière générale, au rang n:
    V*([n,gar]) = n V*([n,occ])=min{V*([n−1,.])
    Seul le cas libre est intéressant, sinon on n'a pas de choix:
    V*([n,lib]) = min{ T([n,lib], G, [n,gar]) V*([n,gar]),
    T([n,lib], A, [n−1, lib]) V*([n−1, lib]) + T([n,lib], A, [n−1, occ]) V*([n−1,occ]) }

On peut ainsi, en utilisant une récurrence sur n, calculer V*

FIG. 2 – **Calcul d'utilité.** *Dans le cas de la recherche d'une place pour aller au cinéma de l'exemple précédent, on peut calculer l'utilité de tous les états par récurrence. On trouve alors que avant la place* 6*, il vaut mieux continuer à avancer car* $V^*([n,gar]) > T([n,.],A,[n-1,lib])V^*([n-1,lib]) + T([n,.],A,[n-1,occ])V^*([n-1,occ])$

4 Apprentissage par Renforcement

Le problème posé ici, toujours dans le cadre des Processus Décisionnel de Markov, est de trouver une politique optimale quand le modèle d'évolution du monde (transition $\mathcal{T}$ et récompense r) n'est pas connu. Les algorithmes classiques de calcul de l'utilité ne peuvent s'appliquer.

La solution apportée par l'apprentissage par renforcement [Sutton, 1988], [Watkins, 1989] est de mettre à jour une approximation de l'utilité après avoir effectivement essayé une action pour connaître l'état suivant. Evidemment, de cette manière, on ne peut mettre à jour l'approximation courante de l'utilité que dans l'état présent du système. On parle alors de mise à jour asynchrone.

Le *Q-Learning* [Watkins, 1989] est un des algorithmes les plus connus. Il utilise une fonction $Q(s,a)$ pour approcher l'utilité que l'on recevrait en effectuant l'action a dans l'état s et en suivant la politique actuelle ensuite. Si, dans l'état s on effectue l'action a et que l'on se retrouve ensuite dans l'état s', la mise à jour de cette approximation se fait par: $Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \times opt_{a'} Q(s',a'))$. Mathématiquement la convergence est prouvée si α décroit lentement vers 0 et si chaque état est visité infiniment souvent.

A cause de cette dernière condition, les algorithmes d'apprentissage par renforcement convergent très lentement. Il faut en effet passer un temps considérable à explorer l'environnement, c'est-à-dire essayer de passer par tous les états le plus grand nombre possible de fois. L'exploitation proprement dite de l'utilité pour en dériver une politique optimale ne pouvant optimalement se faire qu'après cette phase d'exploration. S'il est possible de mélanger ces phases d'exploration et d'exploitation, les algorithmes restent lents...

Ainsi, en plus des limitations inhérentes aux MDP, l'apprentissage par renforcement rajoute la notion de vitesse de convergence, ou d'exploitation de l'apprentissage, qui reste très lente. Le problème des environnements très dynamiques n'en est que plus crucial.

5 Les limitations

Bien que performantes pour gérer les environnements aléatoires et inconnus, les méthodes s'appuyant sur le formalisme des MDP sont limitées par de nombreux facteurs. Nous allons maintenant préciser certaines de ces limites.

Les modèles utilisés doivent avoir un nombre fini d'états, ce qui ne permet pas de modéliser facilement des environnements continus. Certaines techniques de discrétisation "intelligente", d'utilisation de plus proches voisins ou de fonctions d'interpolations permettent de passer outre cette limitation. On se retrouve alors généralement avec un trop grand nombre d'états pour que les algorithmes puissent être utilisés dans la pratique.

Puisque la politique est une fonction d'état, il est nécessaire que l'agent connaisse totalement l'état du monde pour pouvoir choisir son action optimale. Quand on ne connaît pas entièrement l'état, on parle de Processus Décisionnels de Markov Partiellement Observés. Des algorithmes comme WITNESS [Littman, 1994], U-Tree [McCallum, 1996] ou l'utilisation d'observables exhaustifs [Dutech, 2000] permettent de traiter le problème, mais une fois encore au détriment du nombre d'états.

Dans un MDP, le modèle d'évolution du monde comprend à la fois la fonction de transition et la fonction récompense. Il en découle qu'on ne peut chercher une politique optimale que quand ces deux fonctions sont fixées. Cela signifie entre autre que le but actuel ne peut changer (ce qui modifierait la récompense) ou que de nouveaux obstacles ne peuvent pas facilement être pris en compte (cela modifierait les transitions). Ainsi, pour chaque modification mineure du problème actuel, il faut entièrement recalculer une nouvelle politique optimale.

Dans le cas d'un environnment dynamique, comme c'est le cas dans les systèmes multi-agents où les autres agents modifient constammment l'environnement (il leur suffit de s'y déplacer), les MDP sont alors clairement difficiles à utiliser. Une solution parmi d'autres serait d'utiliser l'apprentissage par renforcement pour apprendre le modèle au fur et à mesure qu'il évolue, mais nous avons vu que l'apprentissage était un procédé beaucoup trop lent pour cela.

Il existe encore d'autres limitations mais, comme celles que nous venons d'évoquer, elles ne sont que des variations autour de trois limitations principales des MDP qui sont : *la complexité qui croit exponentiellement avec le nombre d'états et d'actions, la lenteur de la convergence* et *un environnement qui doit être statique*. Les propositions de la section suivante visent à dépasser ces limitations.

6 Apprendre à s'adapter

Plusieurs techniques, utilisées pour les MDP ou dans d'autres domaines de l'I.A. sont à l'origine des idées dévelopées dans cette Section.

- **aggrégation d'états / macro-actions**. On diminue la complexité du modèle en regroupant plusieurs états en un seul [Laroche, 2000] ou en utilisant des actions plus complexes [Parr, 1998], comme par exemple : sortir du batiment par le nord. Les solutions éventuelles sont alors sous optimales, mais plus rapides à obtenir.
- **auto-évaluation.** Capacité pour un agent d'évaluer ses performances actuelles afin de diriger les modifications de ses propres algorithmes ou de la façon d'aborder un problème [Schimdhuber, 1996]. L'agent peut par exemple se baser sur son taux actuel de récompense.
- **transfert de connaissance.** Quand un problème est résolu, l'agent pourrait chercher des informations pertinentes sur la structure et les relations de cause à effet dans son environnement. On peut par exemple essayer de déterminer les sous-buts utiles qui jalonnent la route vers le but principal. L'agent pourrait alors ré-utiliser cette connaissance pour l'appliquer à la recherche de solutions pour des problèmes similaires.
- **adaptation.** Au sens de l'adaptation dans le raisonnement à partir de cas. Connaissant une solution à un problème et les différences entre le problème actuel et l'ancien, comment modifier la solution du premier problème pour l'adapter au nouveau. Malheureusement, dans le cas des MDP où deux environnements très proches peuvent conduire à des politiques très différentes, beaucoup de travail reste à faire.

En combinant toutes ces idées, il devrait être possible d'utiliser les MDP avec plus d'efficacité et en réduisant l'effet des limitations précédentes. Le principe générale de cette technique, qui reste encore largement dans le domaine de la spéculation, serait d'utiliser des modèles d'abord assez grossiers pour trouver rapidement des politiques acceptables. Ces modèles seraient ensuite raffinés, précisés ou modifiés (en terme de détails, de buts, de transitions) avant qu'une nouvelle politique soit adaptée de l'ancienne. Le schéma de la Figure 3 donne une vue intuitive de cette proposition.

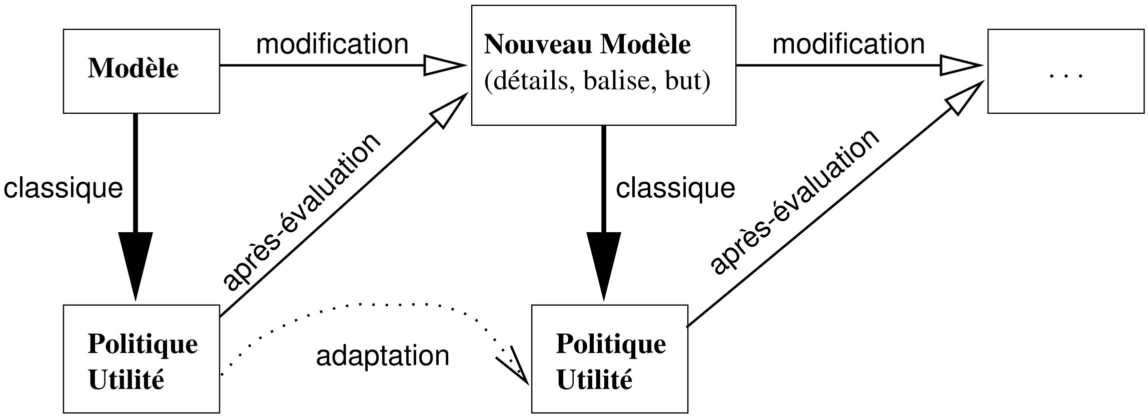


FIG. 3 – **Adaptation dans les MDP.** *On part d'un modèle très grossier de l'environnement d'où il est rapide de tirer une politique. Ensuite ce modèle est modifié par l'agent qui veut améliorer ses performances, prendre en compte des changements dans l'environnement, chercher à atteindre un nouveau but... Le calcul de la nouvelle politique pourrait alors être réalisé plus rapidement que par des méthodes classiques.*

7 Conclusion

Nous avons d'abord expliqué comment nous utilisons les Processus Décisionnels de Markov pour aborder des environnements aléatoires et mal connus. Mais des contraintes fortes font que, en pratique, de nombreux problèmes ne peuvent être résolus en utilisant le formalisme des MDPs. En nous appuyant sur des idées développées actuellement, nous avons essayé de voir comment il serait possible de dépasser ces limitations. Nous n'avons fait ici que défricher un terrain qui reste encore bien vierge.

Références

[Dutech, 2000] A. Dutech. Solving pomdps using selected past events. Dans *ECAI-00*, 2000.

[Ferber, 1995] J. Ferber. *Les Systèmes multi-agents - Vers une intelligence collective*. InterEditions, 1995.

[Laroche, 2000] P. Laroche. *Processus Décisionnels de Markov appliqués à la planification sous incertitude*. PhD thesis, Université Henti Poincaré Nancy I, 2000.

[Littman, 1994] M. Littman. The witness algorithm: solving partially observed markov decision processes. Rapport Technique CS–94–40, Computer Science Dept, Brown University, Providence, Rhode Island, 1994.

[McCallum, 1996] A. McCallum. Learning to use selective attention and short-term memory in sequential tasks. Dans *Fourth International Conference on Simulating Adaptive Behavior*. MIT Press, Cambridge, Massachusetts, 1996.

[Parr, 1998] R. Parr. *Hierarchical control and learning for markov decision processes*. PhD thesis, Computer Science, University of California at Berkley, 1998.

[Puterman, 1994] M. Puterman. *Markov Decision Processes: discrete stochastic dynamic programming*. John Wiley & Sons, Inc. New Yrok, NY, 1994.

[Schimdhuber, 1996] J. Schimdhuber. Simple principles of meta-learning. Rapport Technique IDISIA-69-96, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Lugano, Switzerland, 1996.

[Sutton, 1988] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[Watkins, 1989] C. Watkins. *Learning from delayed reward*. PhD thesis, King's college, Cambridge, 1989.