Apprentissage actif dans les processus décisionnels de Markov partiellement observables

L'algorithme MEDUSA

Robin Jaulmes — Joelle Pineau — Doina Precup

McGill University, School of Computer Science,
3480 University Street, Montreal, QC, Canada, H3A2A7
{rjaulm,jpineau,dprecup}@cs.mcgill.ca

RÉSUMÉ. *Nous cherchons à adapter le cadre des Processus Décisionnels de Markov Partiellement Observables (POMDP) aux réalités de la robotique, pour établir une méthode qui effectue une prise de décision et un apprentissage optimaux lorsque l'agent ne dispose que d'un modèle approximatif d'un environnement non stationnaire. La méthode suppose l'existence d'un opérateur extérieur (oracle), capable d'observer et de révéler l'état caché du POMDP pendant la phase d'apprentissage. Pour résoudre cette problématique de manière approchée, nous proposons l'algorithme MEDUSA, qui confronte le modèle initial à l'expérience directe, et avec l'aide d'un nombre restreint de requêtes, parvient à obtenir rapidement le comportement optimal. Nous montrons comment MEDUSA prend en compte des environnements non stationnaires, et comment il peut s'accomoder de requêtes imprécises.*

ABSTRACT. *We study a problem inspired from robotics in which we want to find an optimal policy to learn a Partially Observable Markov Decision Process (POMDP) when the agent only has an imperfect model of its environment. To help the agent in its task we assume the availability of an external operator (an oracle), that can provide information about the underlying state. We present the algorithm MEDUSA, which improves an initial POMDP model using experimentation through the environment and a minimum number of queries. We also show how MEDUSA handles non-stationary environments and how it can withstand noise in the query answer.*

MOTS-CLÉS : *POMDP, apprentissage actif, modèles non stationnaires, robotique, observabilité partielle, apprentissage par renforcement, MEDUSA.*

KEYWORDS: *POMDP, active learning, non-stationary models, partial observability, reinforcement learning, MEDUSA.*

RSTI - RIA – 1/2007. Prise de décision séquentielle, pages 9 à 33

1. Introduction

La problématique que nous étudions a pour origine la prise de décision optimale pour un robot autonome. Nous supposons que ce robot effectue séquentiellement des actions en se basant sur l'information qu'il reçoit de son environnement par ses capteurs. Ceux-ci ne lui donnent qu'une information partielle sur l'état courant du monde : ainsi, le robot n'a pas directement accès à sa position, il ne peut que percevoir ses alentours. De même, nous nous plaçons dans le cas où les caractéristiques des moteurs et des capteurs sont mal connues. Pire, leur fonctionnement peut varier à cause de phénomènes imprévisibles (luminosité, personnes ou appareils à proximité...). Nous étudions comment un tel robot peut avoir un comportement optimal, et comment il peut apprendre les caractéristiques de son environnement.

Les Processus Décisionnels de Markov Partiellement Observables (POMDP)(Kaebling *et al.,* 1998) forment un cadre qui permet de modéliser les systèmes (agent, environnement) partiellement observables, et d'y effectuer une prise de décision séquentielle optimale. De nombreux algorithmes (Pineau *et al.,* 2003, Poupart *et al.,* 2004, Spaan *et al.,* 2005) ont été proposés pour la planification dans les POMDP. Mais ils nécessitent tous un modèle connu et stable de l'environnement.

Dans de nombreuses applications, un modèle préalable approximatif de l'environnement s'obtient beaucoup plus facilement qu'un modèle préalable exact. C'est pourquoi l'on souhaite avoir à notre disposition un agent capable d'affiner au fur et à mesure les caractéristiques de son environnement par l'expérience directe. Peu de méthodes existent pour apprendre ainsi les paramètres d'un POMDP. Les méthodes de type EM (Chrisman, 1992) sont sujets aux minima locaux. Les approches basées sur l'expérience (McCallum, 1996, Shani *et al.,* 2004, Singh *et al.,* 2003), quant à elles, nécessitent de très larges quantités de données[1], ce qui est incompatible avec un système robotique réel, d'autant plus que ces méthodes posent aussi comme condition un système parfaitement stable.

Puisque nous nous intéressons à des applications de robotique, qui ont des bases réelles, l'état caché du POMDP correspond à une réalité physique mesurable. C'est pourquoi nous envisageons que pendant une période courte au début de l'apprentissage, l'on mette à la disposition de l'agent un opérateur (un GPS, une caméra, un humain), que nous appellerons par la suite *oracle* capable d'identifier, si nécessaire, l'état caché du POMDP. Aidé de cet opérateur, l'agent apprend rapidement les caractéristiques de son environnement avant d'être livré à lui-même.

Au final, on souhaite obtenir un agent capable à la fois d'avoir un comportement optimal en cas d'incertitude sur le modèle et d'y effectuer une exploration intelligente lorsqu'il souhaite l'apprendre. Il doit être capable, avec ou sans l'aide d'un opérateur extérieur, de mettre à jour les paramètres de son environnement au fur et à mesure de ses observations.

---

1. Le nombre d'actions nécessaires est de l'ordre du million, y compris pour des problèmes simples à 2 états.

Dans la section 2 nous exposons le cadre des POMDP. La section 3 présente nos idées sur l'adaptation de l'apprentissage actif aux POMDP et une première méthode pour résoudre cette problématique. Dans la section 4 nous présentons l'algorithme MEDUSA qui est notre deuxième méthode. La section 5 étudie les propriétés de l'algorithme, et la section 6 présente sa performance expérimentale. Nous concluons dans la section 7.

2. Processus Décisionnels de Markov Partiellement observables (POMDP)

Nous présentons ici les POMDP, avec les notations que nous utiliserons dans cet article. Un processus décisionnel de Markov partiellement observable (Kaebling *et al.*, 1998) se compose de :

– un ensemble fini d'**états** S. L'état caractérise à lui seul l'ensemble du système et toute son évolution passée à un instant donné. Dans un POMDP l'état du système n'est pas toujours observable ;

– un ensemble fini d'**actions** A. À chaque cycle l'agent doit choisir l'une d'entre elles. Précisons que seul l'état courant et l'action choisie influencent l'évolution du système et d'éventuels changements d'état ;

– un ensemble fini d'**observations** Z. Après chaque action l'agent reçoit une observation, liée à l'action qu'il vient d'entreprendre et à l'état dans lequel il vient d'arriver ;

– les **probabilités de transition** :
$\{P^a_{s,s'}\} = \{p(s_{t+1} = s' | s_t = s, a_t = a)\}, \forall s \in S, \forall a \in A, \forall s' \in S$;

– les **probabilités d'observation** :
$\{O^a_{s,z}\} = \{p(z_t = z | s_t = s, a_{t-1} = a)\}, \forall z \in Z, \forall s \in S, \forall a \in A$;

– un **facteur de pondération** $\gamma \in [0; 1[$;

– une fonction déterministe de **récompense** $R : S \times A \times S \times Z \rightarrow \mathbb{R}$, telle que $R(s_t, a_t, s_{t+1}, z_{t+1})$ est la récompense immédiate pour la transition correspondante. Cette fonction est uniquement liée à la tâche que l'agent doit accomplir et n'est pas une observation supplémentaire ;

– **un état de croyance initial** $b_0 \in [0; 1]^{|S|}$, qui est la distribution sur les différents états du POMDP à l'instant initial ;

Au début de chaque cycle, l'agent est dans un état $s_t \in S$. Il choisit et exécute une action $a_t \in A$, arrive dans un état inconnu $s_{t+1} \in S$ et observe $z_{t+1} \in Z$, qui est une observation liée à l'état $s_{t+1} \in S$. Les agents qui utilisent des stratégies basées sur les POMDP utilisent la plupart du temps pour leurs décisions l'**état de croyance** $b_t \in [0; 1]^{|S|}$, qui est la distribution de probabilité sur les états cachés possibles compte tenu de l'état de croyance initial b_0 et de l'historique H du processus. Pour suivre l'évolution de l'état de croyance b_t, il suffit d'effectuer, à chaque cycle, une **mise à jour bayésienne** dont l'équation est la suivante :

$$\forall s \; b_{t+1}(s) = \frac{p(z|s,a) \sum_{s0 \in S} p(s|s0,a) b_t(s0)}{\sum_{\{s1,s2\} \in S^2} p(z|s2,a) p(s2|s1,a) b(s1)} \qquad [1]$$

Une **politique** est une fonction qui associe une action à chaque état de croyance possible. Résoudre un POMDP, c'est trouver la politique qui maximise l'espérance du retour :

$$E[\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}, z_{t+1})]$$

Trouver la solution exacte à horizon infini pour un POMDP demande beaucoup de ressources, mais de nombreuses méthodes existent pour trouver des solutions approximatives. Nous utilisons, pour calculer les solutions des POMDP l'algorithme PBVI, qui maximise l'espérance du retour en considérant un échantillon des états de croyance (Pineau *et al.,* 2003). Toutefois, les méthodes que nous proposons pourraient être utilisées avec d'autres algorithmes.

Nous supposons que l'agent connaît la fonction de récompense R, puisqu'elle est directement liée à la tâche que l'on veut lui faire exécuter, qu'il connaît le nombre d'actions, d'observations, et enfin d'états[2], et nous nous concentrons sur l'apprentissage des paramètres $\{P^a_{s,s'}\}$ et $\{O^a_{s,z}\}$. Nous supposons également que l'état de croyance initial b_0 est connu.

Les méthodes classiques pour apprendre les modèles POMDP à partir de l'expérience (Chrisman, 1992) utilisent des méthodes de type EM (Expectation-Maximization), qui alternent entre l'algorithme Baum-Welch (Shatkay *et al.,* 1997), qui détermine les états traversés les plus probables vu le modèle actuel, et un algorithme de maximisation qui trouve les paramètres les plus vraisemblables compte tenu des états traversés. Ce type de méthode est sujet aux minima locaux : la convergence vers le vrai modèle n'est jamais garantie.

D'autres approches utilisent des structures partiellement observables sans modèle (McCallum, 1996, Shani *et al.,* 2004, Singh *et al.,* 2003). Elles sont adaptées aux cas où il est difficile d'utiliser une bonne représentation à base d'états, et lorsque l'on a très peu d'information préalable sur le modèle. En revanche, elles utilisent un nombre extrêmement élevé de cycles (de l'ordre de 10^6) pour apprendre ne serait-ce que des POMDP simples (c'est-à-dire 2-3 états/actions/observations), ce qui les rend impossibles à utiliser pour des agents robotiques réels.

3. Apprentissage actif et POMDP

Nos travaux s'inspirent des techniques d'*apprentissage actif* (Cohn *et al.,* 1996), utilisées pour des tâches de classification dans lesquelles certains échantillons ont des données incomplètes. L'agent a la possibilité d'obtenir les données manquantes pour un petit nombre d'entre eux, et doit en déduire une règle de classification optimale. Les solutions habituellement proposées pour résoudre ce type de problèmes consistent à évaluer le gain d'informations qu'apporte l'obtention d'une donnée manquante pour

2. Le nombre d'actions et d'observations est connu à l'avance quelle que soit l'application. Par contre, connaître le nombre d'états implique une connaissance approfondie de la structure du système et de sa dynamique, c'est donc une hypothèse audacieuse.

chaque échantillon. Ces idées peuvent être appliquées à des systèmes dynamiques tels que les modèles cachés de Markov[3] (Anderson *et al.,* 2005), où la donnée manquante est l'état caché du cycle échantillon.

Nous adaptons ces travaux au cadre des POMDP en prenant en compte plusieurs points. Tout d'abord, par la sélection de l'action, l'agent a une influence sur les états traversés, contrairement au cas du modèle caché de Markov. Il faut donc sélectionner à la fois les actions effectuées et les instants où l'on choisit d'identifier l'état caché. La finalité est également différente. Le but n'est plus d'identifier avec précision tous les états traversés mais de trouver une politique optimale.

Nous allons nous restreindre aux cas où l'état caché est complètement mesurable. Nous supposerons donc qu'il existe un agent extérieur (un *oracle*) capable de l'identifier à la demande. Cette hypothèse audacieuse n'est pas complètement irréaliste : dans la plupart des tâches, il est possible, pour un certain coût, d'accéder avec certitude à l'état caché, à condition d'avoir un observateur extérieur capable de fournir, pendant ou après l'expérimentation, les états traversés par l'agent.

On envisage ainsi, pendant une courte phase de calibrage, de superviser l'agent par un opérateur capable de lui fournir son état. Cette phase devra être la plus courte possible de façon à rendre l'agent autonome rapidement. Ce scénario est réalisable dans le domaine de la robotique, si l'on utilise un agent humain au service de la machine.

Il existe une solution théoriquement optimale pour résoudre la problématique de l'apprentissage actif dans un POMDP.

1) Nous supposons que certains des paramètres du POMDP $U_1 \dots U_k$ ne sont pas connus avec certitude et n'évoluent pas dans le temps.

2) Les paramètres sont fixés à l'instant initial aléatoirement selon les lois de probabilité Π_0^i. Ainsi $p(U_i = x) = \Pi_0^i(x)$.

3) L'agent peut effectuer des **requêtes** pour obtenir en tant qu'observation l'état caché du POMDP.

4) Les requêtes ont un coût, puisque l'on souhaite limiter leur utilisation.

Par application de ces quatre propositions, on crée à partir de notre POMDP original $\{S, A, Z, P, O, R, B_0\}$ le POMDP $\{S', A', Z', P', O', R', B'_0\}$ tel que :

---

3. Un modèle caché de Markov est un modèle POMDP dans lequel une seule action est possible. Ce type de modèle représente une succession d'états partiellement observables reliés les uns aux autres de manière non déterministe.

$$
\begin{aligned}
S' &= S \times [0;1]^K \text{On notera les états } \sigma = \{s, v_1, v_2 \dots v_K\} \\
A' &= \{A; \text{Requête}\} \\
Z' &= \{Z, S\} \\
P'^{a}_{\sigma,\sigma'} &= 0 \text{si } \exists k \in [1;K] v_k \neq v'_k \\
&= \delta(s,s') \text{si } \forall k \in [1;K] v_k = v'_k \text{et } a = \text{Requête} \\
&= P^a_{s,s'} \text{si } \forall k \in [1;K] v_k = v'_k, a \neq \text{Requête}, \forall k P^a_{s,s'} \neq U_k \\
&= v_k \text{si } \forall k \in [1;K] v_k = v'_k \text{et } a \neq \text{Requête et } P^a_{s,s'} = U_k \\
O'^{a}_{\sigma,z} &= 0\delta(s,z) \text{si } a = \text{Requête} \\
&= O^a_{s,z} \text{si } a \neq \text{Requête et } \forall k \in [1;K] O^a_{s,z} \neq U_k \\
&= v_k \text{si } a \neq \text{Requête et } O^a_{s,z} = U_k \\
R'(\sigma, a, \sigma', z) &= R(s,a,s',z) \text{si } a \neq \text{Requête} \\
&= R_q < 0 \text{si } a = \text{Requête} \\
B'_0(\sigma) &= B_0(s)\Pi_0(v_1, v_2 \dots v_K)
\end{aligned}
$$

La résolution de ce POMDP permet d'obtenir la politique d'apprentissage actif optimale. En effet, compte tenu de sa construction, on démontre que l'état de croyance du nouveau POMDP vérifie à tout instant :

$$B(s, v_*) = B(s|v_1, \dots, v_K, B_0, H)\, p(U_1 = v_1, \dots, U_K = v_k|H, \Pi_0) \qquad [2]$$

$B(s|v_1, \dots, v_K, B_0, H)$ est l'état de croyance obtenu par les mises à jour bayésiennes successives correspondant à l'historique H et à l'état initial B_0, lorsque les paramètres incertains valent $v_1 \dots v_K$.

$p(U_1 = v_1, \dots, U_K = v_K|H, \Pi_0$ est la vraisemblance des valeurs $v_1 \dots v_K$ compte tenu de l'historique observé. Cette valeur tend en général[4] vers 1 pour les vraies valeurs et vers 0 pour les autres : ce paramètre est donc un indicateur de l'apprentissage.

Le modèle que l'on obtient lors de cette méthode possède un espace des états infini. Puisqu'à l'heure actuelle aucune méthode ne permet de trouver une politique de POMDP s'appliquant à un espace des états infini, on discrétise chaque variable en

4. C'est lorsque l'historique d'apprentissage observé peut s'expliquer par plusieurs valeurs des paramètres incertains que l'on n'obtient pas la convergence. Cela est en particulier le cas pour des POMDP où certains des paramètres incertains n'ont aucune influence sur la politique optimale.

n niveaux : ainsi, la distribution préalable des valeurs Π_0 est fixée non nulle en un nombre fini de points.

Nous avons analysé la performance de cette méthode sur le problème standard, "Tiger"(Kaebling *et al.,* 1998) que nous modifions de la manière suivante : nous supposons dans notre expérience que nous ne connaissons pas avec certitude la probabilité d'avoir l'observation correcte lorsque l'action "Listen" est exécutée et nous considérons trois valeurs possibles : 0.7, 0.8 et 0.9 (et leur attribuons les probabilités respectives de $1/3$, $1/3$ et $1/3$). Les autres paramètres étant supposés connus.

Pour le POMDP résultant, les méthodes exactes sont incapables de trouver une solution exacte à horizon infini en un temps raisonnable. Toutefois la méthode approximative PBVI (Pineau *et al.,* 2003) est capable de trouver assez rapidement une politique à horizon fini.

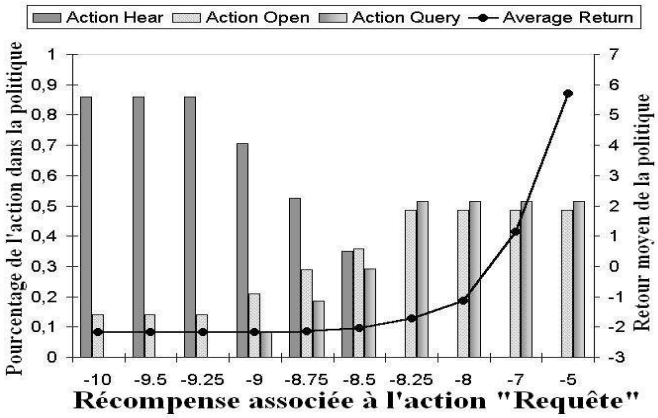


Figure 1. *Résultats expérimentaux sur le problème "Tiger" avec le protocole expérimental décrit en section 3 : on fait varier la valeur de la pénalité associée à l'action "requête". Les barres indiquent le pourcentage de chaque action dans la politique suivie pendant l'exécution, et la ligne indique la performance (le retour moyen) de la solution trouvée*

La figure 1 résume les comportements trouvés et les récompenses qui leur sont associées, en fonction de la pénalité associée à la requête. L'algorithme trouve deux types de politiques solution : le premier type alterne entre l'action requête et l'action optimale[5], et le deuxième type n'effectue aucune requête mais parvient tout de même à un comportement efficace. Pour des valeurs intermédiaires, la politique effectue quelques action "Listen" pour évaluer la valeur du paramètre, puis, selon les résultats, effectue systématiquement des requêtes ou les supprime complètement. Ces résultats montrent qu'il est possible d'avoir une politique raisonnable sans apprendre le modèle exact, et qu'une certaine forme d'apprentissage peut se faire sans effectuer de requête.

5. Dans ce cas, la pénalité associée à une requête est trop faible et la politique consistant à en effectuer après chaque action permet à l'agent de dépasser la performance de la meilleure solution du POMDP initial soit 1,9.

Cette approche ne convient cependant pas à des POMDP ayant un trop grand nombre d'états : en effet, si l'on discrétise chacun des k paramètres incertains en n niveaux, elle multiplie le nombre d'états du modèle initial par n^k. Cela rend vite la recherche de la politique optimale impossible. De plus, il est très difficile de chiffrer la pénalité R_q : celle-ci doit être suffisamment élevée pour motiver l'apprentissage, et suffisamment basse pour que les requêtes soient tout de même envisagées.

4. L'algorithme MEDUSA

Cette section présente l'algorithme MEDUSA, qui s'applique à l'apprentissage actif de modèles POMDP. Contrairement à l'approche optimale présentée dans la section 3, MEDUSA modélise l'incertitude sur le modèle de façon compacte et peut résoudre des problèmes POMDP plus étendus. Nous exposons les techniques qu'utilise MEDUSA pour intégrer l'information préalable sur le modèle et l'expérimentation avec et sans requête. Nous expliquons comment MEDUSA se prête à l'apprentissage d'un POMDP y compris lorsque l'environnement est non stationnaire ou lorsque la précision des requêtes n'est pas de 100%.

L'exploration bayésienne permet d'apprendre les paramètres d'un MDP en modélisant l'incertitude par des distributions de Dirichlet (Dearden *et al.,* 1999, Strens, 2000). L'algorithme que nous présentons peut être vu comme une extension de ces travaux aux POMDP.

4.1. *Le modèle d'incertitude*

Comme indiqué dans la section 2, les probabilités de transition et d'observation des POMDP sont spécifiées par des distributions multinomiales. Les distributions de Dirichlet sont des distributions sur les valeurs que peuvent prendre ces paramètres.

Considérons une distribution multinomiale de dimension N et de paramètres $(\theta_1, \dots \theta_N)$, tels que $\sum_{i=1}^{N} \theta_i = 1$. La distribution de Dirichlet D, caractérisée par les hyperparamètres $(\alpha_1, \dots \alpha_N)$ donne la probabilité d'un ensemble de paramètres multinomiaux θ_i quelconque selon la formule suivante :

$$p(\theta_1 \dots \theta_N | D) = \frac{\prod_{i=1}^{N} \theta_i^{\alpha_i - 1}}{Z(D)}, \text{ où } Z(D) = \frac{\prod_{i=1}^{N} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{N} \alpha_i)} \quad [3]$$

Les paramètres les plus vraisemblables, $\theta_1^* \dots \theta_N^*$ valent quant à eux :

$$\theta_i^* = \frac{\alpha_i}{\sum_{k=1}^{N} \alpha_k}, \forall i = 1, \dots N \quad [4]$$

On a ici $\Gamma(x) = \int_0^\infty t^{x-1} \exp(-t) dt$.

Les distributions de Dirichlet ont pour avantage qu'il est facile de les mettre à jour directement à partir des données expérimentales [6] et qu'il est possible de les utili-

6. Lorsque l'occurence i est observée on augmente le paramètre α_i de 1.

ser pour échantillonner des distributions multinomiales. Pour procéder à cet échantillonnage, il faut considérer N distributions de densité proportionnelle à la fonction Gamma, de paramètres $\alpha_1 \dots \alpha_N$, et les échantillonner[7] ; on obtient alors $x_1 \dots x_N$. La distribution multinomiale que l'on échantillonne a alors pour paramètres :

$$\forall i,\ \theta_i = \frac{x_i}{\sum_{k=1}^{N} x_k}$$

MEDUSA utilise une distribution de Dirichlet pour représenter chaque groupe de paramètres $P^a_{s,*}$ ou $O^a_{s,*}$ incertain et constituer ainsi un **modèle d'incertitude** pour les paramètres incertains du POMDP.

4.2. *Fonctionnement de l'algorithme*

Notre algorithme, appelé MEDUSA pour "Markovian Exploration with Decision based on the Use of Sampled models Algorithm", est une approche qui se fonde sur le raisonnement suivant.

– puisque notre objectif est la recherche de la politique optimale, il n'est pas nécessaire d'apprendre avec précision chacun des paramètres du modèle POMDP.

– chaque requête apporte des informations sur la transition et l'observation qui viennent d'avoir lieu. Ces informations permettent de mettre à jour le modèle d'incertitude. Plus on effectue certaines transitions, plus on augmente dans le modèle la précision de leurs paramètres.

– la période d'apprentissage doit être minimale, ce qui n'est pas le cas lorsque les actions sont aléatoires. On préférera donc utiliser des politiques **potentiellement optimales**. Ce sont les paramètres qui interviennent dans celles-ci qui influent sur la politique optimale.

– plutôt que d'utiliser un seul POMDP pour calculer les politiques potentiellement optimales, on en utilise plusieurs, échantillonnés selon le modèle d'incertitude, afin de prendre en compte le mieux possible les conséquences de l'incertitude de chacun des paramètres.

Le déroulement de l'algorithme est le suivant. Tout d'abord, on construit le modèle d'incertitude : c'est un système de distributions de Dirichlet qui rassemble toute l'information préalable dont on dispose sur notre modèle POMDP. Ses paramètres évoluent au fur et à mesure que l'on accumule de l'expérience avec la procédure d'**apprentissage**. L'agent maintient par ailleurs une **population d'hypothèses POMDP** qui est caractéristique du système de Dirichlet courant. Chaque hypothèse contient ses propres valeurs des paramètres du modèle POMDP, maintient son propre état de croyance, et participe à la politique exécutée.

L'algorithme se déroule ensuite en une succession de cycles. À chaque cycle, l'agent sélectionne l'une des actions recommandées par les hypothèses : l'action $a_i = \pi_i(b_i)$ est choisie avec la probabilité w_i donnée par l'équation 5.

7. On procède selon la méthode décrite dans (Devroye, 1986).

$$\forall i \; w_i = \frac{p_i}{p0_i} \frac{1}{\sum_{k=1}^{n} \frac{p_k}{p0_k}} \qquad [5]$$

p_i est ici la probabilité *actuelle* de l'hypothèse i selon le mosèle d'incertitude courant et $p0_i$ la probabilité qu'elle avait dans le modèle d'incertitude au moment de son échantillonnage.

Les w_i pondèrent l'importance relative de chaque hypothèse. En effet, puisque l'on ne renouvelle pas entièrement la population d'hypothèses à chaque étape (on garde la majorité des modèles d'un instant à l'autre), et que le système de Dirichlet, quant à lui, évolue, on ne peut laisser à chaque modèle une influence égale. C'est pourquoi ces facteurs de pondération sont nécessaires, comme démontré en section 5.1.

Le cycle se poursuit par la réception pour l'agent de l'observation. Il peut alors décider d'effectuer une requête afin d'identifier auprès de l'oracle l'état caché du système. Que la requête soit effectuée ou non, les paramètres correspondants dans le système de Dirichlet sont mis à jour, selon la procédure d'apprentissage détaillée dans la section 4.4.

La population d'hypothèses est régulièrement mise à jour puisqu'à intervalles réguliers on élimine le modèle le moins probable, en le remplaçant par une hypothèse echantillonnée selon le modèle d'incertitude actuel.

À chaque fois qu'une nouvelle hypothèse est échantillonnée, l'agent trouve (dans les limites de temps et de mémoire qui sont les siennes) la meilleure politique qui lui correspond selon une méthode d'approximation de la fonction de valeur (Cassandra, 2004). Cette nouvelle politique contribue alors au choix des actions, ce qui améliore la politique globalement exécutée.

L'algorithme est récapitulé dans le tableau 1 et dans les sections suivantes.

4.3. *La procédure d'initialisation*

L'initialisation du système de Dirichlet est la première étape de notre algorithme. Cette initialisation permet d'**intégrer l'information préalable**.

Il n'est en effet pas nécessaire d'utiliser une distribution de Dirichlet pour chaque distribution inconnue du POMDP.

On peut ainsi **fixer** les paramètres du POMDP considérés comme certains. Dans nos expériences, lorsque le véritable paramètre vaut 0 ou 1, on fait l'hypothèse que cela est su avec certitude. Ainsi, pour toutes les hypothèses échantillonnées, ces paramètres restent fixes.

1) **Initialisation** de la structure de Dirichlet avec l'information préalable (Cf. section 4.3).

2) **Échantillonnage** des hypothèses POMDP (Cf. section 4.1) et évaluation de leurs probabilités p_0 selon l'équation 3.

3) **Calcul** de la politique optimale pour chacune des hypothèses(Cassandra, 2004).

4) **Boucle principale** : exécution de N cycles.

a) **Prise de décision** - sélection de l'action (Cf. sections 4.2, 4.7).

b) **Exécution de l'action**, obtention de l'observation.

c) **Mise à jour** des états de croyance b et β pour chaque hypothèse (Cf équation 1).

d) **Calcul des heuristiques** EntropAlt, GainInfo, Var, et NReq (Cf section 4.5).

e) **Décision de requête** et réglage de la valeur du taux d'apprentissage pour le cycle courant (Cf section 4.5).

f) **Réalisation de la requête** : Le cas échéant, obtention de l'état du système auprès de l'oracle puis mise à jour de l'état de croyance alternatif pour chacune des hypothèses (Cf section 4.4).

g) **Procédure d'apprentissage**. Mise à jour des paramètres du système de Dirichlet (selon les équations 6 et 7 s'il y a eu requête et selon les équations 9 et 11 dans le cas contraire).

h) **Ré-échantillonnage** : Si le cycle courant est un multiple de T_r (période entre deux rééchantillonages), échantillonnage d'une nouvelle hypothèse selon le système de Dirichlet courant, calcul de la politique optimale correspondante, et suppression, lorsque le nombre total de modèles excède Nmax, de l'hypothèse de plus faible probabilité.

Tableau 1. *L'algorithme MEDUSA*

Par ailleurs, il est parfois possible d'utiliser une unique distribution de Dirichlet pour plusieurs ensembles de paramètres[8].

C'est en prenant ces remarques en compte que l'on construit la **structure de Dirichlet** : il s'agit d'un ensemble de distributions de Dirichlet combiné à un ensemble de pointeurs. chaque pointeur correspond à un paramètre du POMDP et indique sa valeur si ce paramètre est certain, ou un indice dans l'une des distributions de Dirichlet lorsqu'il est incertain.

Ainsi, lorsque l'on échantillonne un POMDP, on échantillonne des distributions multinomiales pour chaque distribution de Dirichlet de la structure selon la méthode décrite en 4.1. On consulte ensuite le système de pointeurs. Il indiquera où rechercher chaque paramètre du POMDP, qu'il soit certain ou incertain.

L'ensemble de pointeurs permet également de déterminer quel hyperparamètre mettre à jour lorsque l'on observe une transition ou une observation donnée.

La valeur que nous donnons initialement aux hyperparamètres des distributions de Dirichlet se déduit de l'information préalable dont on dispose. Chaque paramètre est

8. Par exemple, la distribution sur l'ensemble des observations peut être similaire quel que soit l'état ou quelle que soit l'action, et une seule distribution de Dirichlet pour toutes les probabilités d'observation peut parfois suffire.

initialisé à $\alpha_0 \leftarrow \tilde{p} * C_p$ où $\tilde{p}$ est la valeur que l'on présuppose pour le paramètre p, et $C_p \in \mathbb{R}^+$ est la confiance que l'on associe à l'ensemble des paramètres de la distribution correspondante. Dans toutes les expériences de la section 6, on initialise les paramètres incertains avec une confiance de 1 et une distribution uniforme.

4.4. *La procédure d'apprentissage*

Nous étudions à présent le processus de mise à jour du modèle d'incertitude : il intervient à chaque cycle et prend en compte les éventuelles requêtes qui sont effectuées.

Nous utilisons dans cette procédure le concept d'**état de croyance alternatif**, que l'on note β. Il permet de suivre l'information sur l'état de croyance que l'on peut déduire de la dernière requête. Pour chaque modèle, on suit son évolution pour chaque hypothèse selon l'équation 1. De plus, lorsque le résultat d'une requête s_q parvient à l'agent, l'état de croyance alternatif devient $\forall i, s\ \beta_i(s) = \delta(s, s_q)$[9].

Lorsqu'une requête est effectuée, on obtient l'état post-transition s', et on procède aux mises à jour suivantes.

$$\forall s\ \alpha_t(s,a,s') \leftarrow \alpha_t(s,a,s') + \sum_i w_i \beta_i(s) \lambda \quad [6]$$

$$\alpha_z(s',a,z) \leftarrow \alpha_z(s',a,z) + \lambda \quad [7]$$

Ces équations sont une conséquence directe des règles de mises à jour des paramètres des distributions de Dirichlet exposées en 4.1. Ici l'état de croyance alternatif précédant la dernière étape β est utilisé pour mettre à jour les paramètres de transition. Ainsi, si une requête avait été effectuée à l'instant précédent, un seul paramètre est mis à jour à la fois pour les paramètres d'observation et de transition. Plutôt que d'effectuer des incrémentations de 1, on effectuera des incrémentations de λ, où λ est le **taux d'apprentissage**. Plus il est élevé, plus la confiance dans la valeur des paramètres augmentera rapidement, mais puisque leurs valeurs oscillent beaucoup au début de l'apprentissage, les politiques des hypothèses échantillonnées peuvent être trop variables, ce qui diminue la performance de l'algorithme.

Toutefois on peut envisager de se passer de requête. En effet l'information *action-observation* peut être suffisante pour mettre à jour les paramètres des distributions de Dirichlet. On utilise alors **l'état de croyance de transition alternatif** $B_t(s,s')$ qui est évalué selon l'équation 8 ; on met alors à jour les paramètres des transitions selon l'équation 9 [10]. Pour les paramètres correspondant aux probabilités d'observation, les mises à jour se font proportionnellement au nouvel **état de croyance alternatif moyen** $\tilde{\beta}'$ défini par l'équation 10. La mise à jour se fait selon l'équation 11.

---

9. $\delta(i,j)$ vaut 0 quand $i \neq j$ et 1 quand $i = j$.
10. On peut aussi échantillonner un résultat de requête à partir de l'état de croyance et effectuer la mise à jour comme si l'on avait un véritable résultat de requête. Les résultats expérimentaux montrent qu'il n'y a pas de différence notable de performance entre les deux approches.

$$\forall s, s' \; B_t(s,s') = \sum_{i=1}^{n} w_i \frac{[O_i]^a_{s,s'} \, [P_i]^a_{s,s'} \, \beta_i(s)}{\sum_{\sigma \in S} [O_i]^a_{\sigma,z} \, [P_i]^a_{\sigma,s}} \quad [8]$$

$$\forall (s,s') \; \alpha_t(s,a,s') \leftarrow \alpha_t(s,a,s') + \lambda \, B_t(s,s') \quad [9]$$

$$\forall s \; \tilde{\beta}'(s) = \sum_{i=1}^{n} w_i \beta'_i(s) \quad [10]$$

$$\forall s' \in S \; \alpha_z(s',a,z) \leftarrow \alpha_z(s',a,z) + \lambda \, \tilde{\beta}'(s') \quad [11]$$

Lorsque les requêtes sont effectuées en permanence, la procédure d'apprentissage garantit la convergence vers le vrai modèle, comme nous le démontrons en 5.2.

En revanche, cette garantie ne tient plus en leur absence : l'apprentissage sans requête est sujet aux minima locaux. La variance de cet apprentissage étant particulièrement élevée, il devient nécessaire de diminuer la valeur du taux d'apprentissage ; on utilisera un taux 100 fois plus faible. La conséquence est une augmentation du nombre de cycles nécessaire à l'apprentissage.

4.5. *La décision de requête*

Nous exposons les heuristiques que nous utilisons pour décider si une requête doit être effectuée ou non :

– `Var`= $\sum_i w_i (Q(m_i, \Pi(h, m_i)) - \hat{Q})^2$
où $Q(m_i, \Pi(h, m_i))$ est la valeur à laquelle l'action optimale correspond pour chaque hypothèse m_i dans la population courante ; les poids w_i sont ceux définis par l'équation 5 ; $\hat{Q}$ est la valeur moyenne correspondant à l'action optimale $\hat{Q} = \sum_i w_i Q(m_i, \Pi(h, m_i)$. Cette heuristique évalue la variance de l'espérance du retour futur selon le modèle d'incertitude actuel.

– `GainInfo`= $\sum_{k=1}^{n} [w_k \sum_{i,j \in S^2} [B_t(i,j)(\frac{1}{\sum_{k' \in S} \alpha^t_{A,i,k'}} + \frac{1}{\sum_{k' \in Z} \alpha^z_{A,j,k'}})]$
Cette heuristique évalue la quantité d'information que la réponse à une requête pourrait apporter[11]. L'avantage de cette heuristique est qu'elle est égale à zéro quand une requête ne pourrait apporter aucune information nouvelle. Dans ces cas, aucune requête ni apprentissage n'est nécessaire ; il est donc utile d'avoir une heuristique pour les reconnaître.

– `EntropAlt`= $\sum_{s \in S} -[\sum_{i=1}^{N} \beta_i(s)] \log(\sum_{i=1}^{N} \beta_i(s))$
Cette heuristique mesure la quantité d'information qui a été perdue depuis la dernière requête, et à quel point il serait inefficace d'effectuer une mise à jour sans requête.

11. B_t est défini selon l'équation 8. De plus, on considère que $1/\sum_{k' \in S} \alpha^t_{A,i,k'} = 0$ et $(1/\sum_{k' \in Z} \alpha^z_{A,j,k'} = 0)$ lorsque les paramètres correspondant aux transitions (observations) dans l'état i et pour l'action A sont certains.

Utiliser cette heuristique permet à l'agent d'économiser des requêtes, et d'extraire de chaque résultat de requête un maximum d'informations.

– `NReq`. Cette heuristique comptabilise la quantité totale de requêtes effectuées avec succès depuis le début de l'apprentissage.

La requête se fait lorsque la condition suivante est vérifiée :

$$\text{doQ} = (\text{EntropAlt} > \epsilon_1)(\text{GainInfo} > \epsilon_2)((\text{Var} > \epsilon_3)||(\text{NReq} < \text{Nmin}) \quad [12]$$

La première condition permet d'économiser des requêtes lorsque l'état peut être déterminé avec suffisamment de précision par déduction d'une requête antérieure. La deuxième condition permet d'économiser des requêtes lorsque les transitions potentiellement effectuées ont des paramètres suffisamment bien connus. La troisième condition permet d'économiser des requêtes dès lors que des précisions supplémentaires sur les paramètres n'apportent pas de modification significative de la récompense. On incorpore le nombre de requêtes dans le troisième terme puisque la valeur de l'heuristique Var peut être faussée, en particulier au début de l'apprentissage, dans le cas où l'on ne dispose pas d'information préalable, et où les modèles piochés ont des politiques optimales uniformément mauvaises.

Lorsqu'on n'effectue pas de requête le déroulement de la mise à jour dépend du cas dans lequel on se trouve : lorsque `GainInfo`$< \epsilon_2$, on n'effectue pas d'apprentissage. En effet si l'on est dans une partie du modèle où les paramètres sont déjà certains, il n'y aurait de toute façon pas de mise à jour à faire ; et si l'on est dans une partie du modèle où la confiance que l'on a dans les paramètres est très élevée, c'est probablement parce que l'on y a déjà effectué un nombre important de requêtes. Puisque la qualité de l'apprentissage avec requête est supérieure à celle de l'apprentissage sans requête, on n'utilise pas l'apprentissage sans requête car cela ne pourrait que diminuer la qualité du modèle déjà appris. On note que dans le cas non stationnaire il vaut mieux mettre la valeur d'ϵ_2 à zéro car on ne peut alors jamais être certain de la valeur des paramètres appris.

Dans le cas où `EntropAlt`$< \epsilon_1$, il s'avère que l'on peut déterminer avec une quasi-certitude le résultat d'une éventuelle requête. Dans ce cas, puisque la mise à jour sans requête revient au même qu'une mise à jour avec requête, on effectue un apprentissage sans requête avec un taux d'apprentissage plein. Ceci est proscrit dès lors que les résultats de requêtes peuvent comporter des erreurs.

Dans le cas où $(\text{Var} < \epsilon_3)\&(\text{NReq} > \text{Nmin})$, on peut effectuer une mise à jour sans requête des paramètres, mais on utilise un taux d'apprentissage inférieur (le taux d'apprentissage normal est pondéré par 0.01). Cela permet de compenser la grande variance induite par l'apprentissage sans requête. A noter que les deux conditions sont simultanément remplies tard dans l'exécution de l'algorithme, lorsque l'on estime que préciser davantage la valeur des paramètres inconnus n'aura que peu de conséquence sur la fonction valeur, et donc sur la récompense moyenne obtenue par l'algorithme. Il

peut être tout de même avantageux d'utiliser l'apprentissage sans requête pour affiner les paramètres, puisque le même modèle peut dans l'avenir servir à d'autres tâches.

Pour résumer, le déroulement de l'algorithme est le suivant. Dans une première phase, on effectue un apprentissage de bonne qualité, dont le taux d'apprentissage est élevé et où les requêtes sont effectuées à chaque action, excepté lorsque l'on peut en économiser. On passe à la deuxième phase lorsque la variance sur la fonction valeur est en dessous de son seuil. Dans la deuxième phase on n'effectue plus de requête, on se contente de procéder à un apprentissage sans requête, moins précis et plus lent (puisqu'utilisant un taux d'apprentissage plus faible). On finit la deuxième phase lorsque l'on a accumulé suffisamment de données. On passe alors à une troisième phase, où plus aucun apprentissage n'est effectué. Si on le souhaite, on peut alors utiliser une politique plus sûre, telle celle décrite en 4.7.

4.6. *Le cas du modèle non stationnaire*

Nous souhaitons que notre algorithme puisse s'adapter à des environnements non stationnaires, c'est-à-dire des POMDP dont les paramètres varient de manière imprévisible. Dans notre cas, l'algorithme s'adapte très simplement. Il suffit de donner plus de poids à l'expérience récente par rapport à l'expérience ancienne. Notre ajustement est le suivant : à chaque fois qu'un hyperparamètre α est augmenté de la quantité $x\lambda$, on multiplie tous les hyperparamètres correspondant à la distribution multinomiale qui vient d'être modifiée par ν^x, où $\nu \in]0;1]$ est un facteur qui décroît la confiance du modèle. Cette opération n'a aucune influence sur la valeur la plus vraisemblable des paramètres.

$$\forall i = 1, \dots N \ \theta^*_{i,new} = \frac{\nu^x \alpha_i}{\sum_{k=1}^N \nu^x \alpha_k} = \frac{\alpha_i}{\sum_{k=1}^N \alpha_k} = \theta^*_{i,old}$$

Ainsi, plus la mise à jour est ancienne, plus son influence sur la valeur courante est faible. De plus, la confiance d'équilibre, compte tenu de la procédure de mise à jour, a pour expression : $C_{eq} = \lambda \frac{1}{1-\nu}$. Cette confiance, atteinte après un nombre infini de cycles est un indicateur de la confiance globale que l'on a dans l'expérience passée. Plus l'environnement est instable, plus C_{eq} doit être faible.

4.7. *La politique d'exploitation*

Lorsque l'algorithme sélectionne l'une des hypothèses et prend l'action qu'elle recommande (Cf. 4.2), il n'utilise pas toute l'information qui est disponible pour la prise de décision. Le comportement est efficace du point de vue de l'exploration, mais si l'agent est dans une phase où l'apprentissage est fini et où l'important est uniquement de maximiser la récompense, il peut alors suivre une politique dite **d'exploitation**. Ce paragraphe concerne donc un agent qui a un assez bon modèle d'incertitude sur son environnement, qu'il ne cherche pas à l'améliorer dans l'immédiat.

L'idée de la prise de décision sûre est la suivante. Chaque modèle, lorsqu'il est résolu, peut fournir compte tenu de l'historique la valeur associée à chaque action. Soit $V : A \times M \times H \rightarrow R$ la fonction qui retourne, pour un historique $h \in H$, pour un modèle $m \in M$, et pour une action donnée $a \in A$, la valeur qui lui est associée. L'action a^* que l'on choisit est la suivante : si la population de modèles est M_t, l'historique h, et les poids W_m, on a $a^* = \arg\max[\sum_{m \in M} w_m V(a, m, h)]$.

Comme nous le démontrons en 5.1, cette prise de décision est optimale. Toutefois, si l'on considère le gain à long terme, il faut remarquer que la décision sûre ne va jamais "prendre de risque" pour explorer certaines parties du domaine. C'est pour cela que l'on utilise cette prise de décision uniquement quand on **exploite** un modèle appris.

Ainsi, dans chacune de nos expériences, lorsque nous arrêtons le processus d'apprentissage pour évaluer la qualité du modèle appris nous évaluons le retour en exécutant la politique d'exploitation de MEDUSA.

4.8. *Adaptation à des requêtes imparfaites*

Pour parer au cas où la réponse à une requête pourrait être imparfaite, on peut effectuer les corrections suivantes :

– on filtre le résultat des requêtes. Si pour tous les modèles-hypothèses l'état de croyance est nul pour l'état résultat de la requête, on suppose que c'est que l'oracle s'est trompé. Dans ce cas on procède comme si la requête était restée sans réponse ;

– si une requête reste sans réponse, on procède à une mise à jour sans requête à la place ;

– on n'utilise pas l'heuristique EntropAlt pour économiser des requêtes, puisqu'il ne faut pas propager les erreurs dans le résultat des requêtes ;

– on n'utilise pas l'état de croyance alternatif, qui peut dans ce cas être moins fiable que l'état de croyance ;

– on diminue le taux d'apprentissage ;

5. Etude théorique de l'algorithme

Dans cette section nous étudions les propriétés théoriques de MEDUSA. Nous analysons dans un premier temps les limites des politiques utilisées lorsque le nombre de modèles dans la population échantillonnée tend vers l'infini. Puis nous discutons des conditions nécessaires à la convergence de MEDUSA.

5.1. *Analyse de MEDUSA avec une population d'hypothèses infinie*

Nous analysons dans cette section les politiques exécutées par MEDUSA lorsque le nombre de modèles tend vers l'infini. Nous reprenons les notations de la section 2 pour les POMDP. De plus, on désigne par $\mathcal{D}_t$ la structure de Dirichlet correspondant au POMDP à l'instant t du processus. On appelle $\mathcal{M}$ l'ensemble des modèles POMDP

ayant $|S|$ états, $|A|$ actions et $|Z|$ observations. Soit f une fonction de $\mathcal{M}$ dans un sous-ensemble de $\mathbf{R}$. On désigne par $\mathcal{H}$ l'ensemble des historiques (qui contient toutes les séquences possibles d'actions et d'observations).

On note $E(f(m)|\mathcal{D})$ la valeur $\int_{m\in\mathcal{P}} f(m)p(m|\mathcal{D})dm$, où $p(m|\mathcal{D})$ désigne la densité de probabilité de $m \in \mathcal{P}$ selon la distribution $\mathcal{D}$. Il s'agit donc de l'**espérance** de la fonction f lorsqu'on l'applique à un modèle tiré du modèle d'incertitude $\mathcal{D}$.

Théorème 1 : prenons n modèles-hypothèses $m_1 \dots m_n$, échantillonnsés selon les systèmes de Dirichlet $\mathcal{D}_1 \dots \mathcal{D}_n$. Soit $p0_1 \dots p0_n$ les probabilités de chacun des modèles selon leur distribution initiale respective. Soit $p_1 \dots p_n$ les probabilités de chacun des modèles selon la distribution courante, $\mathcal{D}_t$. Soit $f \in \mathcal{M} \rightarrow \mathbf{R}$ la fonction qui retourne, pour un historique, une action, et un modèle donné la valeur 1 si l'action est optimale et la valeur 0 dans le cas contraire. On a alors, pour un historique et une action donnée, lorsque $n \rightarrow \infty$:

$$\forall h \in \mathcal{H}, \forall a \in A, \sum_{i=1}^{n} \frac{p_i}{K\, p0_i} f(m_i) \rightarrow E(f(m)|\mathcal{D}_t) \text{ , avec } K = \sum_{k=1}^{n} \frac{p_k}{p0_k}$$

$$\text{Preuve : } E[\sum_{i=1}^{n} \frac{p_i}{Kp0_i} f(m_i)] = \sum_{i=1}^{n} \frac{p_i}{Kp0_i} E(f(m_i)|\mathcal{D}_i)$$

On effectue une application des techniques d'*échantillonnage par importance*. Il est nécessaire d'utiliser des facteurs correctifs parce que l'on veut évaluer la valeur d'une fonction sur une distribution en utilisant des échantillons qui proviennent d'autres distributions[12]. Donc,

$$\forall i \; E(f(m_i)|\mathcal{D}_i) = \frac{K}{n} \frac{p(m_i|\mathcal{D}_i)}{p(m_i|\mathcal{D}_t)} E(f(m_i)|\mathcal{D}_t)$$

$$\text{Donc } E[\sum_{i=1}^{n} \frac{p_i}{Kp0_i} f(m_i)] = \sum_{i=1}^{n} \frac{1}{n} E(f(m_i)|\mathcal{D}_i)$$

Compte tenu des facteurs correctifs, la fonction appliquée aux échantillons est donc un estimateur non biaisé de l'espérance de la distribution. En conséquence lorsque le nombre d'échantillons tend vers $+\infty$ la moyenne des échantillons converge vers l'espérance de la distribution. □

Corollaire 1 : dans le théorème 1 le membre de gauche désigne la probabilité qu'a l'action a d'être choisie dans la fonction `PriseDeDécision` de MEDUSA si la

12. On remarque que puisque l'on élimine régulièrement les modèles les moins probables, on limite les instabilités, puisqu'on ne se trouve jamais dans le cas où $p_i \rightarrow 0$. De même, comme dans la plupart des cas $p0_i < p_i$ on évite qu'un modèle ait un poids qui domine tous les autres.

situation est $\{h, m_1, \ldots, m_{N_m}, \mathcal{D}_1, \ldots \mathcal{D}_{N_m}, \mathcal{D}_t\}$. Le théorème 1 implique donc que dans la limite d'un nombre de modèles infini, cette probabilité est égale à la probabilité qu'un modèle tiré de $\mathcal{D}_t$ ait a pour action optimale.

Corollaire 2 : si, à la place de la fonction F on utilise la fonction *valeur* V décrite en 4.7, alors la démonstration du théorème 1 prouve que la valeur de l'action a pour l'historique h estimée par MEDUSA tend, pour un nombre infini de modèles, vers son espérance selon le modèle d'incertitude courant. Ainsi l'action optimale sélectionnée par la politique d'exploitation est l'action qui maximise cette espérance ; cela prouve que cette action est optimale.

Nous avons montré qu'utiliser des estimations de valeurs en se basant sur une population de modèles échantillonnés pondérés par leur probabilité, comme c'est le cas dans de nombreux aspects de MEDUSA (politique, heuristiques, apprentissage sans requête) était justifié car ces estimations convergent vers les valeurs moyennes du modèle d'incertitude courant lorsque le nombre de modèles tend vers l'infini.

5.2. *Convergence de MEDUSA*

Nous étudions maintenant les cas dans lesquels la convergence de l'algorithme MEDUSA est garantie. Nous expliquons tout d'abord les conditions nécessaires pour obtenir la convergence vers les vrais paramètres. Puis nous montrons dans quels cas MEDUSA remplit ces conditions.

On appelle **utile** tout paramètre qui *peut* être évalué. Pour un paramètre $P^a_{s,*}$ cela signifie que l'état s peut être atteint avec une probabilité non nulle pour une séquence d'action donnée, et pour un paramètre $O^a_{s,*}$ cela signifie que l'état s peut être atteint en effectuant l'action a avec une probabilité non nulle pour une séquence d'actions donnée. On peut prouver que seuls les paramètres "utiles" ont une influence sur la fonction de valeur et sur la politique. Il n'est donc pas nécessaire pour un algorithme d'apprentissage d'évaluer la valeur des paramètres qui ne sont pas utiles.

Considérons les conditions suivantes : (C1) le modèle que nous apprenons est stationnaire et $\nu = 1$; (C2) nous n'imposons avec la structure de notre modèle d'incertitude aucune condition qui entre en contradiction avec le véritable modèle.

Théorème 2 : si on a (C1), (C2), et qu'une requête est effectuée à chaque étape, l'estimation que l'on a des paramètres utiles converge avec la probabilité 1 vers la valeur correspondante du vrai modèle si, (1) à chaque instant, chaque action a une probabilité non nulle d'être choisie et si (2) on réinitialise le problème un nombre infini de fois.

Preuve : grâce aux règles de mise à jour du système de Dirichlet, décrites par les équations 6 et 7, on a $\forall i,\ \alpha_i = \alpha 0_i + \lambda N_i$, où N_i est le nombre d'occurences du phénomène $< s, a, s' >$ pour un paramètre de type $\alpha_t(s, a, s')$ ou le nombre d'occurences du phénomène $< s', a, z >$ pour un paramètre de type $\alpha_z(s', a, z)$. $\alpha 0_i$ devient négligeable devant λN_i quand N_i tend vers l'infini. Donc l'estimation que l'on a de chaque

paramètre est équivalente à l'estimation de probabilité maximum correspondant aux exemples vus depuis le début de l'expérience.

$$\forall i \in [1 \dots N] \ \theta_i^* = \frac{\alpha_i}{\sum_{k=1}^{N} \alpha_k} \simeq \frac{N_i}{\sum_{k=1}^{N} N_k}$$

.

Or, on sait que l'estimation de probabilité maximum converge toujours vers la valeur réelle lorsqu'il y a un nombre infini d'exemples. Si le paramètre est "utile" et que chaque action a une probabilité non nulle d'être choisie à chaque instant, alors (1) et (2) impliquent qu'il y a pour chaque paramètre un nombre infini d'exemples. □

La version de MEDUSA décrite dans le tableau 1 ne garantit pas que chaque action ait une probabilité non nulle d'être choisie en toutes circonstances. Toutefois, on peut modifier notre algorithme pour offrir cette garantie en établissant un taux d'exploration $0 \leq \epsilon \leq 1$: à chaque instant l'agent a alors la probabilité ϵ d'accomplir une action complètement aléatoire et une probabilité $1 - \epsilon$ d'accomplir l'action optimale. Notre étude expérimentale révèle toutefois que donner une valeur élevée à ϵ nuit grandement à la qualité des politiques exécutées. Mais si l'on souhaite avoir une garantie théorique valide avec un nombre fini de modèles, et quel que soit le POMDP, il faut donner à ϵ une valeur non nulle.

Théorème 3 : si (C1) et (C2) sont vérifiées, si l'on utilise une information préalable qui n'entre pas en contradiction avec le véritable modèle, si l'on utilise les valeurs $\epsilon_1 = 0$, $\epsilon_2 = 0$ et $\epsilon_3 < 0$ pour les paramètres de l'équation 12, et si l'on utilise un taux d'exploration non nul, les paramètres "utiles" estimés par MEDUSA convergent vers leur véritable valeur.

Preuve : si $\texttt{GainInformation} = 0$ effectuer une requête n'a pas d'effet sur le modèle que l'on apprend. De plus, si $\texttt{EntropieEtatAlternative} = 0$ cela signifie qu'effectuer une mise à jour sans requête a exactement les mêmes effets que si l'on utilisait une requête. De plus $\epsilon_3 < 0$ implique qu'on n'utilise pas l'heuristique Var. Donc l'algorithme que l'on utilise a au final le même effet que si l'on réalisait une requête à chaque étape. Puisque le taux d'exploration garantit que chaque action a une probabilité non nulle d'être exécutée à tout instant, le théorème 2 peut donc être appliqué, ce qui prouve la convergence. □

Dans la plupart des environnements POMDP, la structure de récompense est telle que pour toute action et tout historique il existe au moins un modèle qui la recommande [13]. Lorsque c'est le cas, que (C1) et (C2) sont remplies, que la stratégie de requête est celle du théorème 4, et que l'on a un nombre infini de modèles dans notre population, la version de MEDUSA décrite dans le tableau 1 converge vers le véritable modèle.

Preuve : : si cette nouvelle condition sur le POMDP est remplie, alors la probabilité que l'action a soit optimale est non nulle pour tout a quelles que soit les circonstances.

13. Lorsque ce n'est pas vérifié, il existe des actions qui ne sont *jamais* recommandées, pour certains historiques, quelle que soit la valeur des paramètres : cela ne se produit, dans la pratique, presque jamais.

Selon le corollaire du théorème 1, a a donc toujours une probabilité non nulle d'être choisie et il suffit d'appliquer le théorème 2.

6. Etude expérimentale de MEDUSA

Dans cette section nous exposons les expériences que nous avons réalisées avec l'algorithme MEDUSA.

6.1. *Application de MEDUSA a des POMDP de la littérature*

Une suite de problèmes POMDP classiques est disponible pour l'évaluation standard d'algorithmes (Cassandra, 2004). Pour chacun d'entre eux, on construit notre modèle d'incertitude de la manière suivante : tous les paramètres qui correspondent à des transitions déterministes sont connus et certains. De même, on connaît à l'avance les transitions qui sont impossibles. Les autres paramètres sont au départ complètement inconnus. On impose des contraintes entre paramètres pour accélérer l'apprentissage : ces contraintes n'entrent pas en contradiction avec le modèle réel et constituent des hypothèses raisonnables que l'on pourrait avoir sur l'environnement. Dans le tableau 2, on note N_α le nombre total de paramètres dont on cherche à déterminer la valeur, compte tenu des contraintes que l'on impose. Le tableau donne également, pour chaque problème, les caractéristiques qui lui sont propres, ainsi que certains résultats expérimentaux. "Retour Vrai Modèle" indique le retour correspondant à la politique calculée par PBVI (Pineau *et al.,* 2003) pour le véritable modèle, "N.Req." indique le nombre moyen de requêtes nécessaires, expérimentalement, pour atteindre une politique optimale. "N.C.PSR" indique à titre indicatif le nombre de cycles (sans requête, sans information préalable) nécessaires pour apprendre le même modèle avec une méthode de type PSR (résultats tirés de (Wolfe *et al.,* 2005)).

POMDP	|S|	|A|	|Z|	N_α	N.Req.	N.C.PSR	Retour vrai modèle
Paint	4	4	2	6	700	4000	3.27
Shuttle	8	3	5	10	0	1024000	32.80
Network	7	2	4	36	1300	> 2048000	244
1d maze	4	2	2	3	100	-	1.25
4x4 maze	16	4	2	15	200	-	3.73
4x3 maze	11	4	6	16	800	1024000	1.90
Cheese	11	4	7	10	100	32000	3.48
Mini-hall 2	13	3	9	12	75	-	2.71
Hallway	60	5	21	84	450	-	1.02

Tableau 2. *tableau récapitulant les différents POMDP testés. On présente à la fois les caractéristiques de chacun des problèmes et les résultats que l'on obtient*

La figure 2 montre l'évolution du retour pour ces problèmes. On remarque que pour certains la politique est déjà très bonne au début : c'est le cas lorsqu'il n'y a

qu'un nombre limité de paramètres inconnus et que ces paramètres n'influent pas sur la politique solution. Dans certains cas (comme 4x3) les vraies distributions sont uniformes ; l'information préalable que l'on a mis par défaut est très pertinente, mais l'agent doit prendre le temps de le réaliser.

Dans tous les cas, la politique optimale est atteinte à la convergence (excepté dans le problème Hallway, pour lequel la politique est légèrement sous-optimale en fin d'exécution). L'estimation que l'on a des paramètres, quant à elle, converge dans tous les cas où les paramètres sont *utiles* (Cf. section 5.2) vers les valeurs véritables, avec une marge d'erreur d'au plus 5%. En revanche, on constate qu'il n'est pas toujours nécessaire d'avoir une estimation précise de chacun d'entre eux pour atteindre la politique optimale. Lorsque le nombre de requêtes nécessaires est élevé, c'est que la précision qu'il faut atteindre pour atteindre la politique optimale est élevée. Ainsi, pour le problème Shuttle, la politique optimale est la même quelles que soient les valeurs des paramètres incertains, si bien qu'aucune requête n'est nécessaire : tous les paramètres importants sont déterministes. On remarque par ailleurs que le nombre de cycles nécessaires à une méthode de type PSR est en moyenne 100 fois plus élevé que le nombre de cycles nécessaires à MEDUSA. Cette bonne performance est à relativiser puisque les PSR n'utilisent pas d'oracle et sont donc désavantagés.

Par ailleurs, les techniques présentées dans (Wolfe *et al.,* 2005) réalisent uniquement l'apprentissage des paramètres et pas celui de la politique finale, alors que c'est celle-ci sur laquelle se concentre MEDUSA.

6.2. *MEDUSA et les POMDP non stationnaires*

Dans cette section nous présentons des résultats relatifs au comportement de MEDUSA lorsque le modèle est non stationnaire. On règle dans cette série d'expériences le paramètre ν défini en 4.6 à une grandeur inférieur à 1. Les expériences ont lieu dans le domaine Tiger, qu'on a déjà étudié dans la section 3. On considère la probabilité de recevoir l'observation correcte lors de l'action "Listen" p comme incertaine.

On étudie donc le comportement de MEDUSA dans le cas où l'on a appris avec une bonne confiance un modèle et où l'on provoque un brusque changement dans le paramètre p. On peut ainsi voir comment MEDUSA s'adapte à un changement imprévu dans l'environnement.

Sur la figure 3, on voit ce qui se produit lorsque la confiance d'équilibre est faible (100) : l'agent adapte rapidement son comportement au nouveau paramètre même si la variation est importante. Toutefois la faible confiance introduit des oscillations dans les valeurs des paramètres et dans la politique, qui est sous-optimale. Lorsque la confiance est plus importante (1000) on voit que l'agent prend plus de temps pour s'adapter au changement mais que les oscillations sont plus faibles (et la politique à l'équilibre est meilleure).

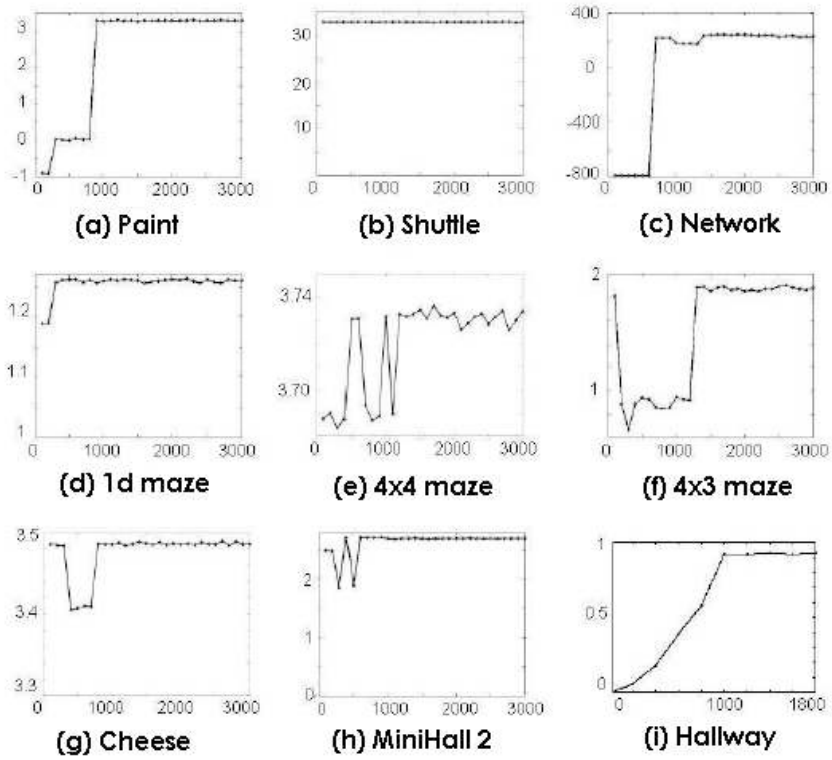
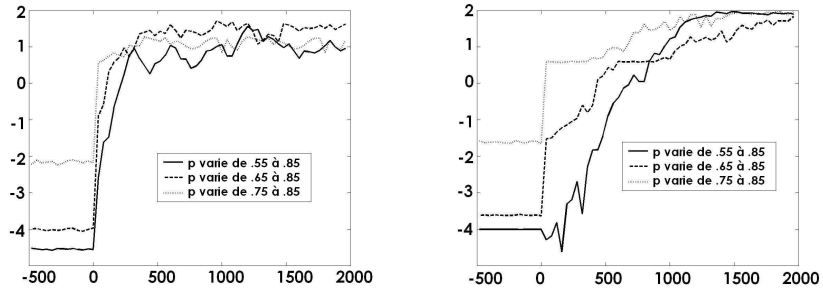


Figure 2. *Evolution de la récompense lors de l'apprentissage des POMDP considérés. Les courbes représentent le retour moyen en fonction du nombre de cycles : celui-ci est évalué en arrêtant l'apprentissage et en exécutant la politique d'exploitation avec les hypothèses actuelles. Le nombre de requêtes qui correspond est donné dans le tableau 2*

6.3. *Performance de MEDUSA face à des requêtes imprécises*

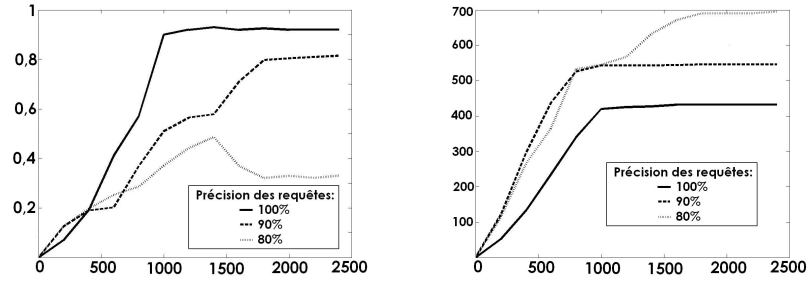
Nous évaluons à présent la robustesse de MEDUSA face à des erreurs dans les réponses aux requêtes. Il est en effet plus réaliste d'envisager que l'oracle peut parfois se tromper. Cela permet d'utiliser des sources d'information plus variées pour jouer le rôle de l'oracle. On peut effectivement envisager d'utiliser comme oracle des capteurs extérieurs tels des caméras, un GPS, ou tout simplement des heuristiques basées sur la séquence d'actions et d'observations. Comme on peut le voir sur la figure 4 sur le problème Hallway, l'algorithme peut supporter un bruit dans la réponse de la requête allant jusqu'à 10% avec une perte minimum de performance. Par contre, le nombre de requêtes nécessaires croît pour l'oracle imparfait. On peut voir que lorsque la précision est inférieure à 0,8 le comportement devient très variable : les mauvaises requêtes font, comme il est illustré sur la figure 4, converger l'algorithme vers des paramètres erronés et une politique sous-optimale.



(a) Confiance à l'équilibre faible (100)

(b) Confiance à l'équilibre élevée (1000)

Figure 3. *Evolution du retour en fonction du nombre de cycles pour le problème Tiger. Il y a un changement soudain dans le paramètre p au cycle 0*



(a) Evolution du retour correspondant au modèle courant en fonction du nombre de cycles

(b) Evolution du nombre de requêtes en fonction du nombre de cycles

Figure 4. *Performance de MEDUSA sur le problème Hallway, avec des requêtes imprécises*

7. Conclusion

Nous avons présenté des stratégies d'apprentissage actif capables de gérer des problèmes d'apprentissage et de prises de décision dans des environnements POMDP où certains des paramètres sont incertains ou peuvent varier de manière inattendue dans le temps. Lors de cet apprentissage actif nous avons supposé que si cela était nécessaire, l'agent pouvait disposer pour l'aider d'un oracle capable d'identifier l'état caché actuel. Nos méthodes doivent permettre à l'agent de n'utiliser cet oracle que pendant son apprentissage et doit pouvoir à terme s'en passer complètement.

Deux méthodes ont été exposées. Notre première méthode construit un nouveau modèle POMDP, dont la résolution apporte la solution de la problématique d'appren-

tissage actif. Peu adaptée à des problèmes étendus, cette méthode a une optimalité garantie.

Notre deuxième méthode, l'algorithme MEDUSA, construit un modèle d'incertitude du POMDP avec des distributions de Dirichlet. Il les utilise pour échantillonner plusieurs modèles-hypothèses, dont il maintient les probabilités respectives. Sa politique tient compte de chacun de ces modèles ; il les améliore graduellement lors de son apprentissage puisqu'il élimine les moins vraisemblables en les remplaçant par d'autres. Nous avons donné des justifications intuitives, théoriques, et empiriques de MEDUSA. Nous avons prouvé sa convergence pour certains cas. Nous avons montré empiriquement comment il était efficace y compris dans des environnements complexes, et comment il était capable de s'adapter à la non stationnarité ainsi qu'aux requêtes inexactes.

Puisque l'algorithme MEDUSA dissocie la prise de décision et l'apprentissage, il est possible de les effectuer sur des processus différents, afin de ne pas pénaliser la prise de décision par des retours de requêtes tardifs ou un temps important passé à calculer la politique optimale d'un modèle échantillonné. Nous souhaitons à l'avenir effectuer cette modification et améliorer la méthode d'apprentissage sans requête afin qu'elle utilise l'algorithme Baum-Welch pour déterminer les états traversés entre deux requêtes ; il faudra alors affiner les heuristiques de décision de requête. Nos travaux futurs porteront par ailleurs sur une application de MEDUSA à des problèmes concrets de robotique.

Remerciements

Les auteurs remercient le Conseil de Recherches en Sciences Naturelles et en Génie du Canada pour son support financier. Les auteurs remercient aussi leurs interlocuteurs du groupe MDPIA pour leur suggestions concernant ce travail, de même qu'Anthony Cassandra pour sa page internet (www.pomdp.org).

8. Bibliographie

Anderson B., Moore A., « Active Learning for Hidden Markov Models : Objective Functions and Algorithms », *ICML*, 2005.

Cassandra A., « www.pomdp.org : Cassandra's POMDP page », 2004.

Chrisman L., « Reinforcement Learning with Perceptual Aliasing : The Perceptual Distinctions Approach », *AAAI*, p. 183-188, 1992.

Cohn D. A., Ghahramani Z., Jordan M. I., « Active Learning with Statistical Models », *J. Artif. Intell. Res. (JAIR)*, vol. 4, p. 129-145, 1996.

Dearden R., Friedman N., Andre D., « Model based Bayesian Exploration », *UAI*, p. 150-159, 1999.

Devroye L., « Non-Uniform Random Variate Generation », 1986.

Kaebling L. P., Littman M. L., Cassandra A. R., « Planning and Acting in Partially Observable Stochastic Domains », *Artificial Intelligence*, vol. 101, n° 1–2, p. 99-134, 1998.

McCallum A. K., Reinforcement learning with selective perception and hidden state, PhD thesis, University of Rochester, Dept. of Computer Science, 1996.

Pineau J., Gordon G. J., Thrun S., « Point-based value iteration : An anytime algorithm for POMDPs », *IJCAI*, p. 1025-1032, 2003.

Poupart P., Boutilier C., « VDCBPI : an Approximate Scalable Algorithm for Large POMDPs », *NIPS*, MIT Press, p. 1081-1088, 2004.

Shani G., Brafman R. I., « Resolving Perceptual Aliasing In The Presence Of Noisy Sensors », *NIPS*, p. 1249-1256, 2004.

Shatkay H., Kaelbling L. P., « Learning Topological Maps with Weak Local Odometric Information », *IJCAI*, p. 920-929, 1997.

Singh S. P., Littman M. L., Jong N. K., Pardoe D., Stone P., « Learning Predictive State Representations », *ICML*, p. 712-719, 2003.

Spaan M. T., Vlassis N., « Perseus : randomized point-based value iteration for POMDPs », *Technical Report IAS-UVA-04-02*, 2005.

Strens M. J. A., « A Bayesian Framework for Reinforcement Learning », *ICML*, 2000.

Wolfe B., James M. R., Singh S., « Learning Predictive State Representations in Dynamical Systems without Reset », *ICML*, 2005.

ANNEXE POUR LE SERVICE FABRICATION
A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :
   RSTI - RIA – 1/2007. Prise de décision séquentielle

2. AUTEURS :
   Robin Jaulmes — Joelle Pineau — Doina Precup

3. TITRE DE L'ARTICLE :
   Apprentissage actif dans les processus décisionnels de Markov partiellement observables

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
   Apprentissage actif dans les POMDP

5. DATE DE CETTE VERSION :
   20 novembre 2006

6. COORDONNÉES DES AUTEURS :

   – adresse postale :

   McGill University, School of Computer Science,
   3480 University Street, Montreal, QC, Canada, H3A2A7

   {rjaulm,jpineau,dprecup}@cs.mcgill.ca

   – téléphone : +1 514 398 7071

   – télécopie : +1 514 398 3883

   – e-mail : {rjaulm,jpineau,dprecup}@cs.mcgill.ca

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :
   LaTeX, avec le fichier de style `article-hermes.cls`,
   version 1.2 du 03/03/2005.

8. FORMULAIRE DE COPYRIGHT :
   Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
   `http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com