Les représentations prédictives des états et des politiques

A. Boularias*
boularia@damas.ift.ulaval.ca

B. Chaib-draa*
chaib@damas.ift.ulaval.ca

*Laboratoire DAMAS
Département d'informatique et génie logiciel, Université Laval
G1K7P4, Québec – Canada

Résumé :
Nous proposons dans cet article une nouvelle approche pour représenter les politiques (stratégies) dans les environnements stochastiques et partiellement observables. Nous nous intéressons plus particulièrement aux systèmes multi-agents, où chaque agent connaît uniquement ses propres politiques, et doit choisir la meilleure parmi elles selon son état de croyance sur les politiques du reste des agents. Notre modèle utilise moins de paramètres que les méthodes de représentation usuelles, telles que les arbres de décision ou les contrôleurs d'états finis stochastiques, permettant ainsi une accélération des algorithmes de planification. Nous montrons aussi comment ce modèle peut être utilisé efficacement dans le cas de la planification multi-agents coopérative et sans communication, les résultats empiriques sont comparés avec le modèle DEC-POMDP (Decentralized Partially Observable Markov Decision Process).

Mots-clés : Incertitude, PSRs, POMDPs, DEC-POMDPs.

Abstract:
We discuss the problem of policy representation in stochastic and partially observable systems, and address the case where the policy is a hidden parameter of the planning problem. We present a new model that generalizes the predictive state representations (PSRs) by introducing tests about the policy. Our approach uses less parameters than the usual policy representation methods, such as the decision trees or the stochastic finite-state controllers. We show how this model can be used efficiently in the cooperative multi-agent planning, and compare it empirically with the Decentralized Partially Observable Markov Decision Process (DEC-POMDP).

Keywords: Uncertainty, PSRs, POMDPs, DEC-POMDPs.

1 Introduction

La planification est certainement l'une des tâches les plus importantes pour n'importe quel agent évoluant dans un environnement dynamique. Un environnement dynamique est un système qui réagit aux actions de l'agent en changeant son état, et en produisant en même temps des observations qui permettent à l'agent de déduire le nouveau état. L'objectif de la planification est d'atteindre certains états désirables du système en choisissant l'action appropriée dans chacun des états intermédiaires. Cependant, dans la plupart des environnements du monde réel, l'état du système est partiellement observable, et l'effet des actions sur l'évolution du système est non déterministe. Les processus décisionnels de Markov partiellement observables (POMDPs : Partially Observable Markov Decision Processes) est un modèle très populaire utilisé pour résoudre ce type de problèmes, et plusieurs algorithmes efficaces pour résoudre les POMDPs ont été développés auparavant. Les POMDPs utilisent une représentation explicite de l'état sous-jacent (caché) sous la forme d'une distribution de probabilité sur tous les états du système. Ce vecteur de probabilité, appelé *l'état de croyance*, dépend directement du nombre d'états spécifiés dans le modèle, ce qui rend les algorithmes de planification extrêmement lents pour les problèmes à très large espace des états. Les représentations prédictives des états (PSRs : Predictive State Representations) [1, 2] est une méthode alternative, récente et prometteuse, permettant de résoudre efficacement ce même type de problèmes. Contrairement aux POMDPs, l'état de croyance dans les PSRs est représenté par un vecteur de probabilités sur des entités complètement observables appelées *tests*. Les

tests sont des séquences finies d'actions et d'observations. De ce fait, l'un des avantages immédiats des PSRs est que l'agent peut apprendre plus facilement le modèle stochastique de transition et d'observation juste en interagissant avec son environnement [3]. De plus, l'agent doit garder uniquement la quantité d'information (reflétée par le nombre de paramètres) suffisante pour prédire l'évolution future du système. Il a été aussi prouvé que dans un type particulier des PSRs, appelés *les PSRs linéaires*, on peut réutiliser tous les algorithmes de planifications développés originalement pour les POMDPs, sans faire une grande modification [4].

Dans plusieurs situations, l'état du système n'est pas le seul paramètre caché du problème, la politique de l'agent peut l'être aussi. En effet, d'un point de vue externe, l'agent est lui même considéré comme un système dynamique qui réagit aux observations qu'il reçoit en produisant des actions, selon une politique qui peut y aller d'une simple fonction réactive : observation→action, à un contrôleur d'états finis stochastique. Les méthodes de recherche de la politique optimale (Policy Search) maintiennent des probabilités sur les différentes politiques, représentant la croyance de l'agent à propos de la politique optimale. Dans les systèmes multi-agents, la politique d'un agent ne peut pas connue des autres, sauf si la communication est possible, et que les agents sont dans un contexte de coopération. La figure 1 montre un problème classique de navigation de robots dans une grille 3×3. Les deux agents, agent 1 and agent 2, choisissent leurs actions de l'ensemble suivant : *Right, Left, Up, Down*, et reçoivent comme observation *W : L'agent vient de toucher un mur* , ou *N : Pas de mur touché*. Si l'objectif de l'agent 1 est de rencontrer l'agent 2, on comprend alors facilement que l'agent 1 doit connaître la politique de l'agent 2 pour pouvoir trouver la meilleure politique qui mène vers un point de rencontre. Mais l'agent 1 ne peut connaître qu'un état de croyance sur les politiques de l'agent 2, représenté par des paramètres probabilistes. La figure 2 montre un ensemble de politiques possibles pour l'agent 2. Intuitivement, l'agent 1 doit maintenir une distribution de probabilité sur toutes ces politiques, et résoudre le problème de la planification en utilisant cette représentation. Malheureusement, cette représentation utilise un nombre de paramètres (probabilités) qui est égal au nombre de politiques, et donc doublement exponentiel en longueur de l'horizon, et en nombre d'observations. Par conséquence, le coût de la solution est très élevé en termes du temps d'exécution et de l'espace mémoire.

D'une manière générale, l'agent fait face à deux types d'incertitudes : d'un coté, l'incertitude sur l'état du système, et de l'autre coté, l'incertitude sur la politique (qui peut être la politique d'un autre agent, ou la politique optimale que l'agent cherche). Le modèle des représentations prédictives des états et des politiques (PSPR : Predictive State and Policy Representations) que nous proposons dans cet article, est une tentative d'unifier ces deux problèmes. En effet, on utilise le même principe, qui est les prédictions, pour représenter les croyances sur les états et sur les politiques. Lorsque la politique n'est pas cachée, ce modèle permet plutôt de représenter la politique d'une manière potentiellement plus compacte que les autres méthodes, telles que les arbres de décision par exemple.

Dans la prochaine section, nous présentons une description étendue des représentations prédictives des états. Dans la section 3, on décrit le modèle PSPR en introduisant la notion des *tests sur la politique*, nous présentons aussi quelques résultats théoriques sur la puissance des PSPRs. On montre par la suite comment ce modèle peut être utilisé dans un contexte de planification multi-agent coopérative et sans communication. On compare alors les résultats empiriques d'un algorithme de planification (Programmation Dynamique en

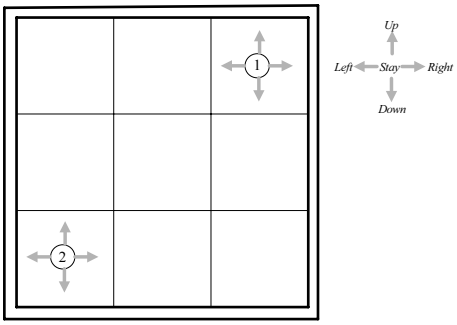

FIG. 1 – Un environnement de navigation de robots dans une grille 3×3.

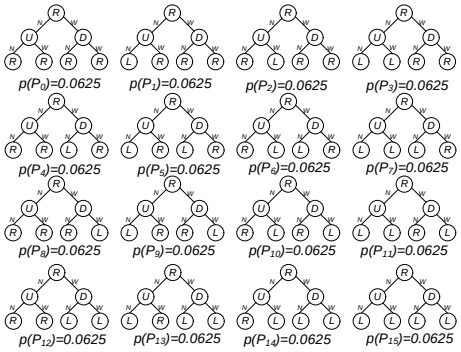

FIG. 2 – Une distribution de probabilité sur l'ensemble des politiques.

l'occurrence) utilisé avec le modèle PSPR et avec le modèle DEC-POMDP [5, 6, 7].

2 Rappel sur les POMDPs et les PSRs

Formellement, un POMDP est de :
- $\mathcal{S}$: Un ensemble fini des états du système.
- $\mathcal{A}$: Un ensemble fini des actions de l'agent.
- Ω : Un ensemble fini des observations de l'agent.
- T : Une fonction de transition et d'observation, $T(s, a, o, s')$ est la probabilité de transiter de l'état s vers l'état s' et d'observer o, lorsqu'on applique l'action a.
- γ : Un facteur d'escompte.
- t : L'horizon de la planification.
- b^0 : La distribution de probabilité initiale sur les états du système.

Notant que les deux derniers paramètres (t et b^0) sont optionnels. Un état de croyance $b^k = (Pr(s_0), Pr(s_1), \ldots, Pr(s_{|\mathcal{S}|-1}))$ est une distribution de probabilité sur les états du système à l'étape k. La propriété de Markov garantit que l'historique du système est encapsulé dans cet état de croyance. En effet, on commence avec un état initial b^0, et à chaque fois que l'agent applique une action a et reçoit une observation o, il met à jours b_k en utilisant la règle de Bayes :

$$Pr(s'|b^k, a, o) = \frac{\sum_{s\in\mathcal{S}} b^k(s)T(s, a, o, s')}{\sum_{s''\in\mathcal{S}} \sum_{s\in\mathcal{S}} b^k(s)T(s, a, o, s'')}$$

Dans le reste de cette section, nous présenterons une brève récapitulation des principes des PSRs tels que décrit dans [2].

L'idée fondamentale des PSRs est de représenter l'état caché du système avec une distribution de probabilité sur tous les scénarios futurs possibles qui peuvent se produirent. Un scénario est une séquence ordonnée des pairs action-observation, qu'on appel [1] *test de l'état* et qu'on dénote par [2] $st = a^1o^1a^2o^2 \ldots a^ko^k$.

La probabilité que le test st réussisse est donnée par $Pr(st) = Pr(o_1 = o^1, o_2 = o^2, \ldots o_k = o^k | a_1 = a^1, a_2 = a^2, \ldots a_k = a^k)$. Dans le cas général, la probabilité de réussir le test st en commençant à l'étape i est $Pr(st|h_i) = Pr(o_{i+1} = o^1, o_{i+2} = o^2, \ldots o_{i+k} = o^k | h_i, a_{i+1} = a^1, a_2 = a^2, \ldots a_{i+k} = a^k)$, ou $h_i = a^0o^0a^1o^1 \ldots a^io^i$ est l'historique complet du système jusqu'à l'instant i. *La matrice*

[1] Cette entité est appelée *test de l'état* au lieu de *test* seulement comme dans [1, 2] , pour pouvoir la distinguer d'un autre type de tests que nous introduisons dans le reste de cet article.

[2] on dénote par a_i (*resp.* o_i) l'action courante (*resp.* observation) à l'étape i, et a^i (*resp.* o^i) une certaine action (resp. observation) donnée de l'ensemble $\mathcal{A}$ (*resp.* $\mathcal{O}$).

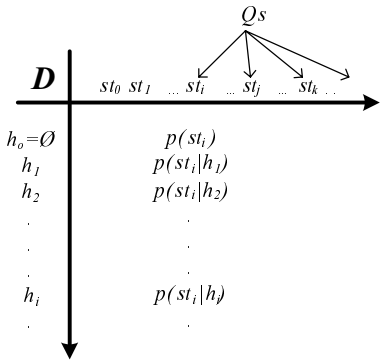

FIG. 3 – La matrice de la dynamique du système.

de la dynamique du système D (Figure 3) est constituée de l'ensemble infini de tous les tests d'état possibles st_i et de tous les historiques h_i possibles. Cette matrice forme un modèle adéquat du système, puisque elle peut être utilisée à toute étape du temps pour prédire le comportement futur du système. Une entrée $D(st_i, h_j)$ de la matrice de la dynamique est la probabilité de réussir le test st_i lorsqu'on commence avec un historique h_j, i.e. $Pr(st_i|h_j)$.

Une caractéristique intéressante de la plupart des systèmes réels est que la probabilité de n'importe quel test st_i dépend uniquement des probabilités d'un certain nombre de tests, appelés *les tests de base*. En d'autres termes, les probabilités des tests de base constituent une statistique suffisante pour le système. Notant que ces tests ne doivent pas être nécessairement effectués pour connaître l'état du système, on a juste besoin de connaître leur probabilités de réussite sans pour autant essayer de les effectuer pour voir s'il réussissent ou pas.

Pour mieux comprendre la notion des tests de base, considérons l'environnement de l'agent 1 dans la figure 1 (et ignorant l'agent 2). Afin de simplifier l'exemple, nous supposons que les actions et les observations sont déterministes. L'agent 1 observe W lorsqu'il à coté d'un mur et il essaie d'avancer dans sa direction, et il observe N dans toutes les autres situations. Ce système contient 9 états (positions possibles de l'agent), mais il peut être représenté uniquement avec 4 tests de base *Left Wall, Right Wall, Up Wall* et *Down Wall*. En effet, si l'agent connaît lesquels de ces tests vont réussir s'il seront effectués et lesquels vont échouer, il pourra alors déterminer exactement dans quelle case de la grille il se trouve. Si on a par exemple Pr(*Left Wall*)=1, Pr(*Right Wall*)=0, Pr(*Up Wall*)=1 et Pr(*Down Wall*)=0, alors on conclue que l'agent se trouve dans la case (0,0) (la première ligne et la première colonne), par conséquence on pourra prédire la réussite ou l'échec de n'importe quel autre test.

On dénote les tests de base par $qs_1, qs_2, \ldots qs_N$, Qs désigne l'ensemble de ces tests. $Pr(Qs|h_j) = (Pr(qs_1|h_j), Pr(qs_2|h_j), \ldots, Pr(qs_N|h_j))$ est le vecteur de probabilité des tests de base, c'est l'équivalent de l'état de croyance pour les POMDPs, on a alors :

$$Pr(st_i|h_j) = f_{st_i}(Pr(Qs|h_j)) \qquad (1)$$

où f_{st_i} est une fonction propre au test st_i. Donc chaque test qui ne fait pas partie de la base, a une fonction associée, indépendante de l'historique, qui permet de prédire sa probabilité de réussite en utilisant uniquement les probabilités de la base Qs.

Après avoir fait l'action a et observer l'observation o, le vecteur $p(Qs|h_j)$ est mis à jours selon la formule :

$$\begin{aligned} Pr(qs_i|h_j ao) &= \frac{Pr(aoqs_i|h_j)}{Pr(ao|h_j)} \\ &= \frac{f_{aoqs_i}(Qs|h_j)}{f_{ao}(Qs|h_j)} \end{aligned} \qquad (2)$$

Si la fonction f_{st_i} est linéaire pour tout st_i, alors le modèle PSR utilisé est dit linéaire, et l'équation (1) peut être simplifiée ainsi :

$$Pr(st_i|h) = Pr(Qs|h) m_{st_i}^T \qquad (3)$$

ou m_{st_i} est le vecteur de poids spécifique au test d'état st_i. Les paramètres d'un PSR linéaire sont :

- Qs, l'ensemble des tests d'état de base.
- $p(Qs|\emptyset)$, les probabilités initiales des tests d'état de base.
- $\forall a \in \mathcal{A}, \forall o \in \mathcal{O} : m_{ao}$, le vecteur de poids du test d'état ao.
- $\forall a \in \mathcal{A}, \forall o \in \mathcal{O}, \forall qs_i \in Qs : m_{aoqs_i}$, le vecteur de poids du test d'état $aoqs_i$ qui est composé du test ao suivi par le test qs_i.

Le cardinal du plus petit ensemble Qs, tel que f_{st_i} est une fonction linéaire, est appelé *la dimension linéaire* du système, qui est aussi le rang linéaire de la matrice D.

Les auteurs de [2] ont prouvé que tout système qui peut être représenté (modélisé) par un POMDP, ou un modèle basé sur l'historique, peut aussi être représenté par un PSR avec un certain nombre de tests de base au plus égal au nombre d'états dans le POMDP. Dans [8], on peut trouver quelque exemples de systèmes qui ne peuvent pas être représentés par aucun POMDP, mais qui peuvent être représentés par des PSRs, et d'autres où le nombre de tests de base est exponentiellement inférieur au nombre d'états dans le POMDP équivalant. Le modèle PSR est alors plus général que les POMDPs, et plus compact. Une autre propriété intéressante est que les paramètres d'un PSR peuvent plus facilement être apprises que les probabilités de transition des POMDPs [3], car on peut toujours connaître si le test réussit ou échoue après quelques étapes du temps, par contre, on ne peut pas vérifier directement l'état sous-jacent du système. Ces propriétés intéressantes sont derrière notre motivation pour généraliser les PSRs de telle sorte que les politique aussi peuvent être représentées par des prédictions sur la réussite ou l'échec de certains tests. Comme on verra dans la section suivante, le problème de la représentation de la politique n'est pas très différents du problème de la représentation des états.

3 Représentation prédictive des politiques

Dans le modèle PSPR que nous proposons ici, on utilise deux types de tests : les tests d'état usuels qu'on a vu dans la section précédente, et *les tests de politique*. Un test de politique pt peut être vu comme le duel du test de l'état, où les actions et les observations sont interchangés. La probabilité qu'un test de politique $pt = o^0, a^1, o^1, a^2, o^2 \dots o^{k-1}, a^k$ réussisse est donnée par $Pr(pt) = prob(a_1 = a^1, a_2 = a^2, \dots a_k = a^k | o_0 = o^0, o_1 = o^1, o_2 = o^2, \dots o_{k-1} = o^{k-1})$. Pour bien comprendre cette notion, imaginons que l'environnement E d'un agent A est lui-même un agent. Donc pour E, A est considéré comme un environnement dont on cherche à connaître la dynamique, et prédire ses comportements futurs. E choisit des actions qu'il applique sur A, qui ne sont rien d'autre que les observations de $\mathcal{O}$, et il reçoit comme observations les actions de $\mathcal{A}$.

La probabilité qu'un test de politique pt réussisse lorsqu'on commence à l'étape i est $Pr(pt|h_i) = Pr(a_{i+1} = a^1, a_{i+2} = a^2, \dots, a_{i+k} = a^k | h_i, o_i = o^0, o_{i+1} = o^1, o_{i+2} = o^2, \dots, o_{i+k-1} = o^{k-1})$. L'historique h_i ici se termine par une action et pas par une observation, et l'étape de temps i désigne l'étape après avoir fait a_i et avant d'observer o_i, on considère aussi que tous les historiques commencent par l'observation fictive o^*. La politique de l'agent peut être représentée par une matrice P (figure 4), construite en considérant l'ensemble infini de tous les historiques possibles (lignes), et tous les tests de politique possibles. Cette matrice est équivalente à la politique de l'agent, même si cette dernière n'est pas stationnaire. Une entrée $P(pt_i|h_j)$ définit la probabilité que l'agent choisisse les actions du test pt_i, sachant que l'historique actuel

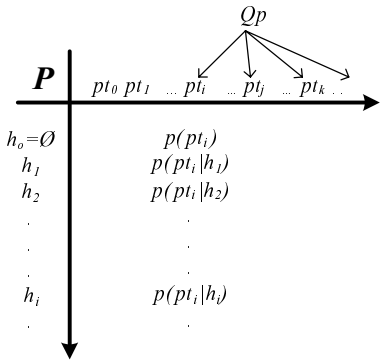

FIG. 4 – La matrice de la politique.

est h_j, et que les observations futures seront celles de pt_i. On définit aussi l'ensemble des tests de politique de base $Qp = \{qp_1, qp_2, \ldots qp_M\}$, ces tests sont suffisants pour déterminer la probabilité de n'importe quel autre test de politique pt_i :

$$Pr(pt_i|h_j) = f_{pt_i}(p(Qp|h_j)) \tag{4}$$

Tel que f_{pt_i} est la fonction associée à pt_i, et $Pr(Qp|h_j) = (Pr(qp_1|h_j), Pr(qp_2|h_j), \ldots, Pr(qp_M|h_j))$.

La fonction de mise à jours des tests de politique de base est donnée par :

$$Pr(qp_i|h_j oa) = \frac{Pr(oaqp_i|h_j)}{Pr(oa|h_j)}$$

$$= \frac{f_{oaqp_i}(Qp|h_j)}{f_{oa}(Qp|h_j)} \tag{5}$$

Les paramètres des repésentations prédictives des politiques sont :
– Qp, les tests de politique de base.
– $(Qp|\emptyset)$, les probabilités initiales des tests de base.
– $\forall a \in \mathcal{A}, \forall o \in \mathcal{O} : f_{oa}$, la fonction associée au test de politique oa.
– $\forall a \in \mathcal{A}, \forall o \in \mathcal{O}, \forall qp_i \in Qp : f_{oaqp_i}$, la fonction associée au test de politique $oaqp_i$ composé du test oa suivi par le test qp_i.

Si la fonction f_{pt_i} est linéaire, les paramètres devient alors :
– Qp, les tests de politique de base.
– $(Qp|\emptyset)$, les probabilités initiales des tests de base.
– $\forall a \in \mathcal{A}, \forall o \in \mathcal{O} : m_{oa}$, le vecteur des poids du test de politique ao.
– $\forall a \in \mathcal{A}, \forall o \in \mathcal{O}, \forall qp_i \in Qp : m_{oaqp_i}$, le vecteur des poids du test de politique $aoqp_i$.

En utilisant ces paramètres, on peut mettre à jours la probabilité du test qp_i après avoir observé l'évènement oa par :

$$p(qp_i|h_j oa) = \frac{p(Qp|h_j)m^T_{oaqp_i}}{p(Qp|h_j)m^T_{oa}} \tag{6}$$

L'ensemble des paramètres des tests d'état et des tests de politique forment les paramètres du modèle PSPR. Notant que ces deux ensembles sont séparés et utilisés indépendamment l'un de l'autre, car on peut bien représenter les états avec les tests d'état et les politique avec des arbre de décision par exemple, comme on peut bien représenter les politiques avec des tests de politique et les états avec un POMDP par exemple. La relation qui peut potentiellement exister entre ces deux types de test fera l'objet d'une futur investigation.

Les théorèmes suivants permettent de comparer les PSPRs avec quelques autres modèles.

Theorem 1. *Une politique d'un MDP (Markov Decision Process) peut être représentée par un PSPR utilisant au plus le même nombre de paramètres.*

Démonstration. Dans les MDPs, les états sont complètement observables, On a donc $\mathcal{O} = \mathcal{S}$. Une politique π est une fonction de $\mathcal{S}$ vers une distribution de probabilité sur les actions de A, tel que $\pi(s, a)$ est la probabilité que l'agent choisisse l'action a dans l'état s. La politique du MDP est donc représentée avec $|\mathcal{S}||\mathcal{A}|$ paramètres. Dans le modèle PSPR, les tests de politique sa, au nombre $|\mathcal{S}||\mathcal{A}|$, sont suffisants pour décrire π si on considère que

$p(sa|h) = \pi(s,a)$. On peut voire facilement que ces deux représentations sont équivalantes. □

Theorem 2. *Une politique d'horizon fini pour un POMDP peut être représentée par un PSPR utilisant au plus le même nombre de paramètres.*

Démonstration. Une politique d'horizon fini t pour un POMDP est un arbre de décision déterministe (à condition de connaître l'état de croyance initial). Cet arbre de décision contient exactement $\frac{(|\mathcal{A}||\mathcal{O}|)^{t+1}-1}{(|\mathcal{A}||\mathcal{O}|)-1}$ noeuds. La matrice P correspondante à cette politique a $\frac{(|\mathcal{A}||\mathcal{O}|)^{t+1}-1}{(|\mathcal{A}||\mathcal{O}|)-1}$ lignes et $\frac{(|\mathcal{A}||\mathcal{O}|)^{t+1}-1}{(|\mathcal{A}||\mathcal{O}|)-1}$ colonnes, donc le rang de P ne peut pas être supérieur à ce nombre, un PSPR peut utiliser les tests de politique formant les colonnes linéairement indépendantes pour décrire cette politique. □

Theorem 3. *Une politique décrite par un contrôleur d'états finis stochastique peut être représentée par un PSPR utilisant au plus le même nombre de paramètres.*

Démonstration. (sketch) Un contrôleur d'états finis stochastique est un tuple $\langle \mathcal{Q}, \psi, \eta \rangle$, tel que $\mathcal{Q}$ est un ensemble fini d'états du contrôleur, ψ est une fonction définie de $\mathcal{Q}$ vers une distribution de probabilité sur $\mathcal{A}$, *tel que* $\psi(q,a)$ est la probabilité de choisir l'action a dans l'état q du contrôleur. η est une fonction de transition, $\eta(q,a,o,q')$ est la probabilité que le prochain état du contrôleur soit q' lorsque le dernier état a été q, et la dernière action effectuée est a et la dernière observation est o. Si on remplace $\mathcal{Q}$ par $\mathcal{S}$ et on échange $\mathcal{A}$ avec $\mathcal{O}$, on obtient exactement une description d'un POMDP, donc on peut utiliser la même preuve de [2] pour prouver que le rang linéaire de P ne peut pas être supérieur à $|\mathcal{Q}|$. Donc le nombre de tests de politique de base dont on a besoin est au plus $|\mathcal{Q}|$ tests. □

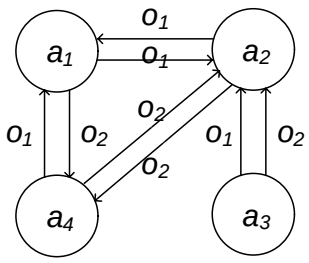

FIG. 5 – Un contrôleur d'états finis déterministe à 4 états qui peut être complètement décrit avec 2 tests de politique seulement.

Les fonction ψ et η dans le contrôleur représenté dans la figure 5 sont déterministes, chaque état est étiqueté par une action. Ce contrôleur contient 4 états, mais il peut être exactement décrit avec les deux tests de politique : $pt_1 = o_1 a_1$ et $pt_2 = o_2 a_2$. On peut vérifier que les réponses à ces deux tests sont suffisantes pour déterminer l'état du contrôleur. Si on a par exemple $Pr(pt_1) = 1$ et $Pr(pt_2) = 1$ on déduit alors qu'on est dans l'état étiqueté par a_4.

Afin de pouvoir comparer empiriquement les performances du modèle PSPR avec les autres modèles, nous avons choisi le problème de la planification multi-agent coopérative pour être la première application des PSPRs, ce choix est motivé par le fait que la représentation des politiques sous l'incertain est une difficulté inhérente à ce genre de problèmes.

4 PSPR versus DEC-POMDPs

Les DEC-POMDPs (Decentralized Markov Decision Processes), proposés récemment par Daniel S. Bernstein *et al.* [5], sont une généralisation des POMDPs aux systèmes multiagent. À chaque étape de temps, chaque agent i fait une action a_i et reçoit une observation o_i et une récompense immédiate r, qui est la même pour

tous les agents bien qu'ils peuvent choisir des actions différentes, ceci les incitent à choisir des politiques individuelles coopératives. Les agents ne peuvent pas communiquer entre eux des informations à propos des actions qu'ils choisissent ou des observations qu'ils reçoivent. L'objectif de la planification dans les DEC-POMDPs consiste donc à trouver la politique jointe optimale, qui est composée de plusieurs politiques individuelles, une pour chacun des agents.

Un DEC-POMDP pour n agents est un tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, t, p_0 \rangle$, tel que :
- $\mathcal{S}$ est un ensemble fini d'états.
- $\mathcal{A}$ est un ensemble fini d'actions pour chaque agent, les agents partagent le même ensemble $\mathcal{A}$ d'actions individuelles. $\mathcal{A}^n$ est l'ensemble des actions jointes.
- $\mathcal{R}(s, a_1, a_2, \ldots, a_n)$ est la fonction de la récompense immédiate.
- $\mathcal{O}$ est un ensemble fini d'observations pour chaque agent, les agents partagent le même ensemble $\mathcal{O}$ d'observations individuelles.
- $\mathcal{T}(s, a_1, a_2, \ldots, a_n, o_1, o_2, \ldots, o_n, s')$ est une fonction de transition et d'observation.
- t est l'horizon de la planification.
- γ est le facteur d'escompte.
- b^0 est l'état de croyance initial.

On désigne par q_i^t une politique à horizon t pour l'agent i, et par $q^t = (q_1, \ldots, q_n)$ une politique jointe à horizon t pour tous les agents. $q_{-i}^t = (q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_n)$ est une politique d'horizon t pour tous les agents sauf l'agent i. on a donc $q^t = \{q_{-i}^t, q_i^t\}$. On utilise Q_i^t pour désigner l'ensemble des politiques q_i^t et Q_{-i}^t pour les politiques q_{-i}^t

Les états et les politiques dans les DEC-POMDPs sont représentés par un état de croyance b_i pour chaque agent i, qui contient une distribution de probabilité sur les états, et une autre distribution sur les politiques jointes q_{-i} des autres agents, car l'agent i ne connaît pas exactement quelles sont les politiques suivi par les autres agents.

Dans les DEC-POMDPs, on a deux fonctions de valeur. La première est la valeur d'une politique jointe q dans un état s :

$$V(s, q) = R(s, A(q)) + \gamma \sum_{o \in \mathcal{O}^n} \sum_{s' \in \mathcal{S}} T(s, A(q), o, s') V(s', q(o)) \quad (7)$$

tel que $A(q)$ la première action jointe qui se trouve à la racine de l'arbre q, o est une observation jointe, et $q(o)$ la politique jointe qui reste dans l'arbre q après l'observation o. On définie aussi la fonction de valeur d'une politique individuelle q_i selon l'état de croyance b_i par :

$$V(b_i, q_i) = \sum_{s \in \mathcal{S}} \sum_{q_{-i} \in P_{-i}} b_i(s, q_{-i}) V(s, \{q_{-i}, q_i\}) \quad (8)$$

Pour trouver la politique jointe optimale à partir de ces formules, les auteurs de [6] ont proposé l'opérateur de la programmation dynamique pour les DEC-POMDPs :

1. Étant donnés les ensembles Q_i^{t-1} des politiques d'horizon $t-1$ pour chaque agent i.
2. Pour chaque agent i, générer à partir de Q_i^{t-1}, l'ensemble Q_i^t de toutes les politiques d'horizon t.
3. Pour chaque agent i, élaguer les politiques complètement dominées. Une politique q_i est dite complètement dominée si et seulement si :
$$\forall b_i, \exists q_i' \in Q_i^t : V(b_i, q_i') > V(b_i, q_i) \quad (9)$$
4. Retourner l'ensemble Q_i^t des politiques optimales pour chaque agent i.

Le problème le plus important dans l'opérateur de la programmation dynamique est l'élagage des politiques dominées, car on doit considérer tous les états de croyance b_i possibles pour chaque agent, ceci peut être fait avec un programme linéaire, mais l'exécution de tels programmes est relativement importante dans ce cas, car le

nombre de variables (états, et politiques) évolue exponentiellement en la taille de l'horizon et en nombre d'observations. Les auteurs de [7] avaient proposé d'utiliser une méthode approximative pour déterminer les politiques dominées. Au lieu de vérifier l'équation 9 pour tous les états de croyance, on peut se limiter à un petit ensemble de points de croyances. Cependant, le nombre de paramètres utilisés pour représenter les probabilités sur les politiques jointes dans chaque état de croyance est toujours important, car on a une probabilité par politique jointe. Dans nos expérimentations, nous avons implémenté une version modifiée de l'algorithme PBDP [7], la seule différence est que nous sélectionnons aléatoirement les points de croyances utilisés pour l'élagage, sans tenir en considération le fait que ces points soient accessibles ou pas.

Nous proposons ici une solution à ce problème qui est basée sur les représentations prédictives. On verra que les états de croyance sur les politiques jointes sont plus compactes lorsqu'on utilise des tests au lieu d'énumérer explicitement toutes les politiques.

Les actions et les observations dans les tests d'état tels qu'on les a vu jusqu'à maintenant sont remplacées par des actions et des observations jointes. Donc, un test st devient $st = a^1o^1a^2o^2 \dots a^ko^k$, avec $a^i \in A^n$ et $o^i \in O^n$.

Puisque on a dans ce cas deux types de politiques, à savoir les politiques individuelles et les politiques jointes (qui ne sont qu'une collection de politiques individuelles, une par agent), on doit alors utiliser deux types de tests de politique, les tests de politique individuels, et les tests de politique joints, qui sont de la forme $pt = o^0a^1o^1 \dots a^k$, avec $a^i \in A^n$ et $o^i \in O^n$.

L'état de croyance b_i pour un agent i est constitué des trois vecteurs :
– Qs : les probabilités des tests d'état.

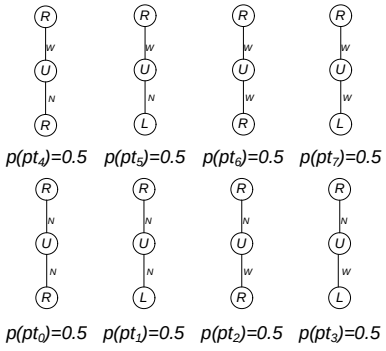

FIG. 6 – La représentation d'un état de croyance sur les politiques dans les PSPRs.

– Qp : les probabilités des tests de politique joints, elles représentent la croyance de l'agent i sur les politiques des autres agents.
– Qp_i : les probabilité des tests de politique individuels, elle déterminent une politique pour l'agent i.

Puisqu'on utilise des politiques déterministes (des arbres de décision), les probabilités de Qp_i sont toutes des 1 ou des 0. On a donc un arbre de décision par vecteur Qp_i, et un vecteur Qp_i par arbre.

Si on considère que les récompenses immédiates font partie des observations, alors on peut utiliser l'équation de Bellman suivante pour résoudre ce problème, avec n'importe quelle technique de programmation dynamique utilisée pour les POMDPs (ou les DEC-POMDP)[4] :

$$V(b_i, q) = \sum_{r \in \mathcal{R}} Pr(A(q)r|b_i)r + \gamma \sum_{o \in \mathcal{O}^n} Pr(A(q)o|b_i)V(\zeta(b_i, A(q), o), q(o)) \quad (10)$$

tel que $\mathcal{R}$ est l'ensemble des récompenses, ζ est la fonction de mise à jours (équation 1 en utilisant les paramètres m_{ao} et m_{aoqs_i}), $A(q)r$ est un test d'état composé de la première action jointe de q et de l'observation (récompense) r.

Pour bien montrer l'apport de l'utilisation

des représentations prédictives des politiques, on a choisit des environnements de test où la représentation prédictive des états est équivalente à la représentation POMDP, i.e. les tests d'état sont linéaires et leur nombre est égal au nombre d'états de l'environnement, donc le fait d'utiliser les PSRs ici n'apporte aucun gain en temps de calcul, puisque ils ne permettent pas de compresser l'espace des états. On va focaliser notre discussion dans ce qui suit uniquement sur les représentations prédictives des politiques.

Initialement, à l'étape d'horizon 1, l'ensemble des politiques Q_i^1 pour un agent i, est représenté par :
- $Qp_i^1 = \mathcal{A}$: L'ensemble des tests de base, qui est formé de toutes les actions individuelles possibles.
- $Pr(Qp_i^1)$: Le vecteur des probabilités initiales des tests de base.
- Les paramètres m_{oa} et m_{oaqp} ne sont pas utilisés ici car l'horizon est 1, donc on ne fait pas de mise à jours des probabilités $Pr(Qp_i^1)$.

Chaque politique de Q_i^1 correspond à une instance de $Pr(Qp_i^1)$. Les paramètres de la politique qui consiste à faire l'action a_i sont donc $Pr(a_i) = 1$ et $Pr(a_j) = 0, \forall a_j \neq a_i$.

L'ensemble Q^1 de toutes les politiques jointes d'horizon 1 est représenté de la même manière que les ensembles Q_i^1, il suffit de remplacer les actions de $\mathcal{A}$ par des actions de $\mathcal{A}^n$.

À l'étape d'horizon $t > 1$, on génère Q_i^t, l'ensemble de toutes les politiques d'horizon t pour l'agent i, à partir de l'ensemble Q_i^{t-1}. Les paramètres de Q_i^t sont :
- $Qp_i^t = Qp_i^{t-1} \bigcup_{a \in \mathcal{A}, o \in \mathcal{O}, q_i^{t-1} \in Qp_i^{t-1}} \{aoq_i^{t-1}\}$.
- $Pr(Qp_i^t)$.
- $m_a(a_j o_j q_i^{t-1}) = 1$ pour $a = a_j$, $m_a(a_j o_j q_i^{t-1}) = 0$ pour $a \neq a_j$.
- $m_{aoq_i^{t-1}}(a_j o_j q_i^{t-1}) = 1$ pour $ao = a_j o_j$, $m_{aoq_i^{t-1}}(a_j o_j q_i^{t-1}) = 0$ pour $ao \neq a_j o_j$.

Les tests de politique de base qu'on a considéré ici sont des séquences qui se terminent avec une action ou une politique. On a choisi d'utiliser des tests de la forme aoq_i^{t-1} (de profondeur 2) plutôt que de la forme conventionnelle $aoao \ldots a$ (de profondeur t) car ça nous permet de réduire le nombre de tests de base utilisés. En effet, si le nombre de politiques par horizon est borné (par le nombre de points de croyance dans notre algorithme) et l'horizon t est suffisament grand, alors on aura besoins de plus de tests de base de profondeur t que de politiques. Par contre, on a besoins de seulement $|\mathcal{A}||\mathcal{O}||Q_i^{t-1}|$ de tests de base de la forme aoq_i^{t-1} générés exhaustivement à partir de Q_i^{t-1}, au lieu des $|\mathcal{A}||\mathcal{O}|^{|Q_i^{t-1}|}$ politiques (arbre) générées dans le modèle DEC-POMDP.

La prochaine étape consiste à élaguer les politiques individuelles qui sont complètement dominées dans tous les points de croyance. On génère d'abord l'ensemble de toutes les politiques jointes Q^{t-1}, cet ensemble est décrit de la même façon que Q_i^{t-1}, avec $a \in \mathcal{A}^n$, $o \in \mathcal{O}^n$, et $q^{t-1} \in \mathcal{Q}^{t-1}$. Les états de croyance pour l'agent i sont donc des vecteurs $b_i = (Pr(Qp^t), Pr(Qs))$. $Pr(Qp^t)$ est une distribution de probabilité sur les tests de politique aoq^{t-1} (c'est une distribution car ces tests sont évènements disjoints), et $Pr(Qs)$ est un vecteur de probabilités sur les tests d'état qs. Pour chaque politique q_i^t, on redistribue les probabilités uniquement sur les tests joints qui sont compatibles avec q_i^t. Un test aoq^{t-1} est dit compatible avec q_i^t si et seulement si l'action de l'agent i dans l'action jointe a est la première action (la racine de l'arbre) de q_i^t, et la politique de l'agent i dans q^{t-1} est la politique qui reste dans q_i^t après l'observation o.
On définie aussi la fonction de valeur de q_i^t, dans un état de croyance b_i, par :

$$V(b_i, q_i^t) = \sum_{aoq^{t-1} \in Qp^t} Pr(aoq^{t-1}|b_i) V(b_i, aoq^{t-1})$$

$$V(b_i, aoq^{t-1}) = \sum_{r \in \mathcal{R}} p(ar|b_i)r$$
$$+\gamma Pr(ao|b_i)V(\zeta(b_i, a, o), q^{t-1})$$

La fonction ζ permet de mettre à jours les probabilité de b_i selon les paramètres m_a et $m_{aoq_i^{t-1}}$ pour les tests de politique de base, et les paramètres m_{ao} et m_{aoqs_i} pour les tests d'état de base.

Cette fonction de valeur est utilisée dans l'équation 9 pour déterminer les politiques complètement dominées et les éliminer de l'ensemble Q_i^t. Le coût du calcul de la valeur d'une politique q_i^t, dans un état de croyance b_i, est de l'ordre de $|\mathcal{A}||\mathcal{O}||Q_i^{t-1}|N$ pour un le modèle $PSPR$, et de $|\mathcal{A}||\mathcal{O}|^{|Q_i^{t-1}|}N$ pour un POMDP. N est le nombre d'états, qui est égal au nombre de tests d'états dans notre cas. La Figure 6 illustre la représentation PSPR d'un état de croyance sur les politiques qui est équivalente à la représentation DEC-POMDP dans la figure 2. Le nombre de paramètres est réduit de 16 à 8. Cette représentation utilise la forme conventionnelle des tests, i.e. les tests se terminent avec des actions et pas avec des politiques.

5 Résultats Expérimentaux

Nous avons testé les performances de l'algorithme de la Programmation Dynamique utilisant le modèle DEC-POMDP et PSPR sur deux problèmes cités dans la littérature [6]. Le premier problème est *Le tigre et la princesse*, où deux agents se trouvent devant deux portes, une à droite et une à gauche. Derrière l'une des portes se trouve une princesse, et derrière l'autre se trouve un tigre. Le problème contient donc deux états. Deux observations avec perturbation sont possibles : Entendre le tigre à gauche, ou l'entendre à droit. L'objectif des agents est d'ouvrir la porte de la princesse. Les actions possibles sont : écouter, ouvrir la porte à gauche, ou ouvrir la porte à droite.

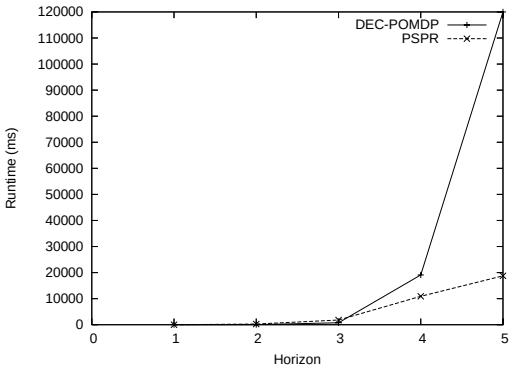

FIG. 7 – Le temps d'exécution en fonction de l'horizon, avec le problème du tigre.

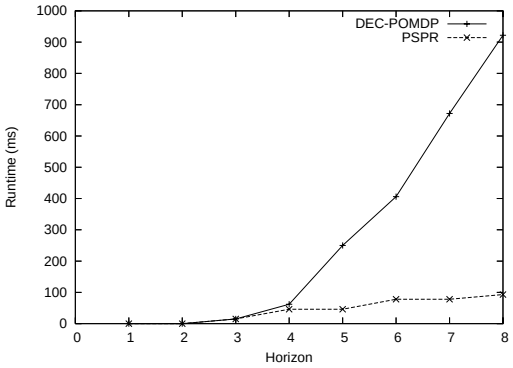

FIG. 8 – Le temps d'exécution en fonction de l'horizon, avec le problème du canal.

Le deuxième problème est celui du *Canal de Communication*, où deux agents échangent des messages à travers un canal partagé, et essaient d'éviter les collisions. Ce problème contient 4 états, selon que chacun des agents a ou n'a pas de message à envoyer, deux observations par agent : collision ou pas de collision, et deux actions par agent : envoyer ou ne pas envoyer un message. On a utilisé le même nombre de tests d'état de base que d'états réels dans ces deux problèmes.

Les figures 7 et 8 montrent le temps d'exécution de l'algorithme en fonction de l'horizon. Ces résultats confirment l'avan-

Le tigre	t=1	t=2	t=3	t=4
DEC-POMDP	-2	-4	5.19	4.80
PSPR	-2	-4	5.19	4.80

Le canal	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8
DEC-POMDP	1	2	2.90	3.89	4.79	5.69	6.59	7.49
PSPR	1	2	2.99	3.80	4.79	5.60	6.50	7.49

TAB. 1 – Les valeurs retournées dans les modèles DEC-POMDP et PSPR avec le problème du *tigre* et le problème du *canal*.

tage d'utiliser les représentations prédictives des politiques par rapport aux représentations nominales. On remarque que le temps d'exécution avec le modèle PSPR est polynomial, quasiment linéaire, alors que le temps d'exécution avec le modèle DEC-POMDP est exponentiel. L'étape qui consomme le plus du temps dans cet algorithme est celle de l'élagage des politiques dominées, et c'est précisement cette étape qui a été améliorée significativement dans le modèle PSPR.

6 Conclusion et Travaux futurs

Dans la plupart des systèmes du monde réel, l'incertitude de l'agent ne porte pas uniquement sur l'état du système, mais bien aussi sur les politiques. Ce problème devient plus crucial dans le cas des systèmes multi-agents. Dans cet article, nous avons proposé une méthode originale qui permet de représenter les politiques et les états en utilisant le même principe, qui est la prédiction. L'avantage de ce modèle, appelé PSPR, est que l'agent utilise uniquement la quantité d'information minimale et suffisante pour représenter sa croyance. Comme première application des PSPRs, on a implémenté un algorithme de programmation dynamique proposé pour résoudre le problème de la planification multi-agent coopérative, et comparé les résultats obtenu de cet algorithme lorsqu'il utilise la représentation standard DEC-POMDP pour modéliser les politiques, et lorsqu'il utilise la représentation prédictive. Ces résultats nous permettent de confirmer que les PSPRs sont un modèle prometteur. Dans les travaux futurs, on essayera de trouver d'autres applications de ce modèle, telles que l'apprentissage par renforcement , qui peut être vu comme une recherche dans l'espace des politiques, ou bien les algorithmes de planification pour le cas mono-agent. On étudiera aussi plus profondement les propriétés théoriques de ce modèle, et comment on peut exploiter efficacement les liens qui existent entre les tests d'état et les tests de politique.

Références

[1] M. Littman, R. Sutton, S. Singh, Predictive representations of state. *Advances in Neural Information Processing Systems 14 (NIPS'02)*. pp. 1555-1561, 2002.

[2] S. Singh, M. James, M.R. Rudary, Predictive state representations : A new theory for modeling dynamical systems. *Uncertainty in Artificial Intelligence : Proceedings of the 20th conference (UAI'04)*. pp. 512-519, 2004.

[3] S. Singh, M. Littman, N. Jong, D. Pardoe and P. Stone, Learning Predictive State Representations, *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*. pp. 712-719, 2003.

[4] M. James, S. Singh and M. Littman, Planning with Predictive State Representations, *Proceedings of the International Conference on Machine Learning and Applications (ICMLA'04)*, pp. 304-311, 2004.

[5] D. Bernstein, N. Immerman and S. Zilberstein, The complexity of decentralized control of markov decision processes, *Journal of Mathematics of Operations Research*, Vol. 27, Num. 4, pp. 819-840, 2002.

[6] E. A. Hansen, D. S. Bernstein and S. Zilberstein, Dynamic programming for partially observable stochastic games, *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI'04)*, pp. 709-715, 2004.

[7] D. Szer and F. Charpillet, Point-based Dynamic Programming for DEC-POMDPs, *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI'06)*, pp. 304-311, 2006.

[8] M. James, Using Predictions for Planning and Modeling in Stochastic Environments, *Thèse de doctorat*, Université de Michigan, 2005.