Programmation dynamique à base de points pour la résolution des DEC-POMDPs

Daniel Szer — François Charpillet

Equipe MAIA
INRIA-LORIA, B.P.239
54506 Vandœuvre-lès-Nancy, France
{szer,charp}@loria.fr

RÉSUMÉ. *Nous présentons un nouvel algorithme de planification pour la construction de systèmes multi-agents réactifs et situés pouvant se modéliser par des processus de décision de Markov décentralisés (DEC-POMDP). Cet algorithme est fondé sur la programmation dynamique à base de points. Il est dérivé de techniques de programmation dynamique optimale utilisées pour résoudre des jeux stochastiques partiellement observables(POSG) et des techniques d'approximation utilisées pour résoudre des POMDP mono-agents. Nous montrons pour la première fois qu'il est possible de déterminer un ensemble d'états de croyance multi-agent pertinents, et nous montrons comment ce calcul permet ensuite d'éviter le recours à la programmation linéaire très couteuse dans le cas multi-agent. Nous détaillons une version exacte et une version approximative de notre algorithme, et nous montrons son efficacité sur un exemple de la littérature.*

ABSTRACT. *We present a novel planning algorithm for building reactive and situated multi-agent systems based on the theory of decentralized Markov decision processes (DEC-POMDPs). The algorithm is a synthesis of multi-agent dynamic programming for partially observable stochastic games (POSGs), and point-based approximations for single-agent POMDPs. We are able to show for the first time that it is possible to determine a set of useful multi-agent belief states, and we show how this computation permits to avoid the linear programming part of current dynamic programming algorithms. We derive both an optimal and an approximated version of our algorithm, and we show its efficiency on a test example from the literature.*

MOTS-CLÉS : *contrôle optimal décentralisé, DEC-POMDPs, planification*

KEYWORDS: *optimal decentralized control, DEC-POMDPs, planning*

1re soumission à *JFSMA'06*, le 13 juillet 2006.

1. Introduction

Nous présentons une nouvelle approche de planification pour la résolution des problèmes de Markov distribués (DEC-POMDP). Nous nous intéressons plus particulièrement à des environnements multi-agents distribués par nature, comme par exemple la nanorobotique, où la communication entre les agents n'est souvent pas possible [SHI 05], ou comme les réseaux de communication, où la communication elle-même fait partie des paramètres à optimiser [ALT 00]. La résolution optimale de tels problèmes est particulièrement dure [BER 02], ce qui motive la recherche d'algorithmes efficaces.

Depuis peu, des algorithmes de programmation dynamique [HAN 04] et de recherche heuristique [SZE 05b] ont été proposé pour résoudre le problème DEC-POMDP sous sa forme générale de manière optimale. Il a été montré également que l'exploitation de structures particulières peut réduire la complexité du pire cas de la solution optimale [BEC 04]. Il existe par ailleurs un certain nombre d'approches approximatives, qui sont souvent basées sur la théorie des jeux [EME 04], la descente de gradient [PES 00] ou d'autres méthodes d'optimisation locale. L'avantage de ces approches est leur relative simplicité de résolution, l'inconvénient principal étant le manque de garanties sur l'optimalité de la solution retournée, puisque ces approches n'ont souvent pas de lien théorique fort avec les approches optimales.

L'apport de notre travail est en deux parties. D'abord, nous établissons une approche formelle de programmation dynamique multi-agent à base de points, ce qui inclut une analyse de l'état d'information dont peut posséder un agent dans un environnement multi-agent décentralisé. Ensuite, nous montrons comment cet algorithme optimal peut être approximé pour permettre la résolution de problèmes plus larges. Nous testons l'algorithme sur un problème de la littérature, et nous pointons finalement vers des directions de recherche future possible.

2. POMDPs décentralisés

Nous basons notre approche sur le formalisme des processus de décision de Markov décentralisés, une extension des POMDPs au cas multi-agent. L'accent est mis sur l'exécution du système, ce qui veut dire que la planification peut être centralisée, mais que les agents exécutent leurs politiques de manière décentralisées.

2.1. *Le formalisme DEC-POMDP*

Nous introduisons le modèle DEC-POMDP conformément à [BER 02]. Un DEC-POMDP à n agents est donné sous forme d'un n-uplet $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O}, T, p_0 \rangle$, où

– $\mathcal{S}$ est un ensemble fini d'états,

– $\mathcal{A}$ est un ensemble fini d'actions,

– $\mathcal{P}(s, a_1, \ldots a_n, s')$ est une fonction de probabilités de transition,
– $\mathcal{R}(s, a_1, \ldots a_n)$ est une fonction de récompense,
– Ω est un ensemble fini d'observations,
– $\mathcal{O}(s, a_1, \ldots a_n, o_1, \ldots o_n)$ est une fonction de probabilités d'observation,
– T est l'horizon du problème,
– p_0 est la distribution initiale sur les états.

Nous soulignons que le modèle DEC-POMDP est très expressif et permet la modélisation d'un grand nombre de systèmes multi-agents discrets. Le modèle inclut en particulier la communication entre agents sous forme de symboles d'observation particuliers. Résoudre un DEC-POMDP pour un horizon fini T et une distribution d'états initiale p_0 revient à trouver un ensemble de n *politiques* qui maximisent l'espérance de la récompense du système $E[\sum_{t=0}^{T-1} \mathcal{R}(s_t, (a_1, \ldots a_n)_t, s_{t+1})|p_0]$.

2.2. *Politiques*

La prise de décision dans un environnement partiellement observable ne peut se faire qu'en fonction des observations reçues, et il a ainsi été montré que les politiques peuvent être représentées sous forme d'arbres de décision (figure 1). Nous notons q_i un arbre de décision et Q_i un ensemble d'arbres pour l'agent i. Le super-ensemble contenant les ensembles de politiques pour tous les agents sauf l'agent i est noté $Q_{-i} = (Q_1, \ldots, Q_{i-1}, Q_{i+1}, \ldots, Q_n)$. L'approche de programmation dynamique que nous allons développer dans les parties suivantes consiste à générer de manière incrémentale un ensemble de politiques utiles pour chaque agent.

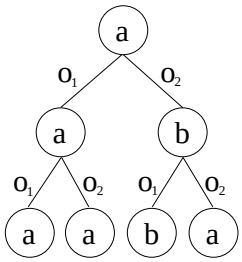

Figure 1. *Un arbre de décision pour un problème à horizon 3, avec 2 observations (o_1 et o_2) et 2 actions (a et b) possibles.*

2.3. *États de croyance*

Un concept primordial dans la prise de décision dans l'incertain est celui d'un *état d'information*, c'est-à-dire de l'ensemble des informations que l'agent possède sur le

système à tout moment. Il a été montré que dans le cas des POMDPs mono-agents, la distribution de probabilités sur les états sous-jacents, conforme aux observations reçues, constitue un état d'information suffisant pour la résolution optimale du problème. On parle alors d'*état de croyance*.

Dans la théorie du contrôle décentralisé ainsi que dans la théorie des jeux, une estimation de l'état sous-jacent du système n'est en général plus suffisante pour permettre la résolution optimale du problème. Il a en effet été montré que même dans le cas des systèmes totalement observables par tous les agents, on est confronté à un problème de coordination : la prise de décision optimale d'un agent nécessite en général une anticipation de la décision de tous les autres agents [CLA 98].

Plusieurs auteurs ont proposés une extension de la notion d'état de croyance au cas multi-agent, et nous adoptons une définition introduit par [NAI 03], puis affiné par [HAN 04]. Un *état de croyance multi-agent* est une distribution de probabilités sur les états du système et le comportement futur de chaque agent, c'est-à-dire leurs politiques probables.

Définition 2.1 (État de croyance multi-agent). *Un état de croyance multi-agent b_i pour l'agent i est une distribution de probabilités sur les états du système et les politiques de tous les agents restants : $b_i \in \Delta(\mathcal{S} \times Q_{-i})$.*

Étant donné un état de croyance multi-agent, le problème de choisir une politique optimale correspondante, que l'on qualifie souvent de politique de "meilleure réponse", devient un problème classique et essentiellement mono-agent. L'approche de programmation dynamique multi-agent revient donc à déterminer, pour chaque agent et sur l'ensemble de l'espace des états de croyance, les politiques de "meilleure réponse" [HAN 04]. Ceci est un problème d'optimisation linéaire continu.

On peut montrer que la majeure partie de l'espace de croyance ne sera jamais visité, et qu'il y a même des croyances qui sont strictement impossibles. Nous argumentons que l'approche de programmation dynamique de [HAN 04] est insatisfaisante dans le sens ou elle ne permet pas d'exclure ces états de croyance et de concentrer l'effort de calcul sur les croyances les plus probables. Nous montrons pour la première fois qu'il est possible d'énumérer explicitement les seuls états de croyance possibles, et nous introduisons une approche de programmation dynamique alternative se basant exclusivement sur les croyances potentiellement possibles.

2.4. *Fonctions de valeur*

La fonction de valeur multi-agent reflète l'évaluation de *politiques jointes*. Une politique jointe δ est un vecteur de n politiques $\delta = (q_1, \ldots, q_n)$, une pour chaque agent. Évaluer une politique jointe δ pour un état s revient à effectuer un calcul de programmation dynamique

$$V(s,\delta) = \sum_{\mathbf{o}\in\Omega^n} P(\mathbf{o}|s,\delta)\left[\sum_{s'\in\mathcal{S}} P(s'|s,\delta,\mathbf{o})V(s',\delta(\mathbf{o}))\right] \quad (1)$$

où $\mathbf{o} = (o_1,\ldots,o_n)$ dénote une observation jointe et $\delta(\mathbf{o})$ représente la politique jointe des sous-arbres après observation de $\mathbf{o}$. Pour une fonction de valeur multi-agent globale, nous introduisons la fonction de valeur pour l'agent i comme

$$V_i(b_i,q_i) = \sum_{s\in\mathcal{S}}\sum_{\delta_{-i}\in Q_{-i}} b_i(s,\delta_{-i})V(s,\langle\delta_{-i},q_i\rangle) \quad (2)$$

où δ_{-i} représente un vecteur de politiques pour tous les agents sauf i et $\langle\delta_{-i},q_i\rangle$ représente donc une politique jointe pour les n agents.

2.4.1. *Programmation dynamique pour DEC-POMDPs*

L'algorithme de programmation dynamique pour DEC-POMDPs proposé par Hansen et al. [HAN 04] consiste en deux phases qui sont appliqués en alternance, à savoir l'énumération exhaustive de l'ensemble des politiques possibles, et l'élagage des politiques entièrement dominées. Une politique est dite *dominée* si elle n'est jamais utile, c'est-à-dire s'il elle ne constitue jamais une "meilleure réponse" pour aucun état de croyance. L'opérateur de programmation dynamique, montré figure 2, garde, à chaque

Entrée : les ensembles de politiques $Q_1^{t-1},\ldots,Q_n^{t-1}$ à horizon (t-1).

1) Pour tout agent i, l'opérateur produit d'abord l'ensemble *exhaustif* de politiques Q_i^t pour l'horizon t à partir de l'ensemble Q_i^{t-1} des politiques à horizon (t-1). Les arbres de politiques contenues dans Q_i^{t-1} vont donc devenir les sous-arbres dans Q_i^t.
2) L'opérateur va ensuite *élaguer* les politiques *dominées*. Une politique q_i est dominée si elle peut être remplacée par une autre politique $\tilde{q_i}$ qui est au moins aussi performante que la première, et ceci pour chaque état de croyance possible. Une politique dominée peut être identifiée par le programme linéaire suivant : la politique q_i peut être élaguée de Q_i^t si

$$(\forall b_i)(\exists\tilde{q_i}\in Q_i^t\backslash\{q_i\}) \quad \text{tel que} \quad V_i(b_i,\tilde{q_i}) \geq V_i(b_i,q_i)$$

Sortie : les ensembles de politiques $Q_1^t,\ldots,Q_n^t$ à horizon t.

Figure 2. *L'opérateur de programmation dynamique multi-agent*

itération, et donc pour chaque horizon de problème, un ensemble de politiques utiles par agent. Après T applications, il retourne les ensembles de politiques utiles pour

l'horizon du problème. Une politique jointe optimale $\delta^* = (q_1^T, \ldots, q_n^T)$ peut alors être déterminée comme suit :

$$\delta^* = \underset{\delta^T \in Q_1^T \times \cdots \times Q_n^T}{\arg\max} \sum_{s \in \mathcal{S}} p_0(s) V(s, \delta^T) \tag{3}$$

3. Programmation dynamique multi-agent à base de points

L'opérateur de programmation dynamique à base de points que nous allons proposer dans la partie suivante adresse deux points faibles de l'approche de programmation dynamique présentée plus haute [HAN 04]. D'une part, nous sommes intéressés à éviter la résolution coûteuse d'un programme linéaire pour identifier les politiques dominées. D'autre part, nous voulons éviter de considérer des régions dans l'espace des croyances qui sont peu probables ou simplement impossibles à atteindre. Nous décrivons par la suite l'approche de programmation dynamique à base de points (PDBP) pour les POMDPs mono-agents.

3.1. *Programmation dynamique à base de points pour POMDPs*

Il a été montré pour la première fois par Lovejoy que la fonction de valeur du POMDP peut être approximé en discrétisant l'espace des croyances en une grille de régions [LOV 91]. Au lieu de déterminer la fonction de valeur sur un espace continue, sa valeur est déterminée aux seul extrémités de la grille. Au lieu d'identifier des politiques dominées sur un espace continue, il suffit alors de garder les meilleures politiques aux points de la grille, ce qui veut dire que le programme linéaire est remplacé par un simple problème de maximisation discret. La valeur de tout état de croyance se trouvant entre les extrémités de la grille peut alors être approximé par interpolation. La qualité de cette approximation dépend évidemment de la résolution de la grille.

Pineau et al. ont proposé récemment une approche plus flexible, se basant sur des *points de croyance* qui ne sont alors plus distribués uniformément dans l'espace, mais aux endroits les plus intéressants pour le contrôle du système [PIN 03]. Ces endroits peuvent être déterminés par une simulation de trajectoires markoviennes, c'est-à-dire une génération stochastique de croyances possibles. Étant donné une croyance b, une action a et une observation o, on peut générer une nouvelle croyance b' en utilisant la formule de Bayes :

$$b'(s') = \frac{\sum_{s \in S} b(s) \Big[\mathcal{T}(s, a, s') \mathcal{O}(s, a, o, s') \Big]}{P(o|b, a)} \tag{4}$$

Le dénominateur $P(o|b, a)$ est le facteur de normalisation habituel. A partir de la croyance initiale b_0, il est donc possible de générer des états de croyance pour tous les horizons du problème.

3.2. *Programmation dynamique multi-agent à base de points exacte*

Nous allons montrer maintenant qu'il est possible de générer des états de croyance multi-agent pertinents, ce qui constitue le cœur de notre algorithme de programmation dynamique. Contrairement au cas mono-agent, où la simulation de l'évolution du système suffit pour déterminer une distribution possible sur ses états, le cas multi-agent nécessite en plus la considération des distributions sur les politiques des agents. Nous différentions donc la croyance de l'agent i sur les états du système au moment t, que nous notons $b_i^{\mathcal{S},t}$, et la croyance qu'il possède sur les politiques d'un autre agent j, que nous notons $b_i^{j,t}$. Un état de croyance multi-agent complet pour un système à n agents se note donc $b_i^t = (b_i^{\mathcal{S},t}, b_i^{1,t}, \ldots, b_i^{i-1,t}, b_i^{i+1,t}, \ldots, b_i^{n,t})$.

Dans la théorie des jeux, une distribution sur des politiques éventuelles est souvent appelé *stratégie mixte*. Il est connu que l'utilité d'une stratégie pour un agent i dépend en général de l'espace des stratégies mixtes de son adversaire. Bien que nous nous plaçons dans un cadre coopératif et non compétitif, l'observabilité partielle, et donc l'incapacité de connaître avec sûreté l'état interne des autres agents, nous contraint de considérer plusieurs alternatives possibles sur leurs politiques.

Comme nous l'avons décrit plus haut, la programmation dynamique pour DEC-POMDPs génère, à l'itération t, l'ensemble des politiques utiles Q_i^t pour les t derniers pas de l'exécution. Déterminer l'utilité d'une politique $q_i^t \in Q_i^t$ revient donc à répondre à la question suivante : quelles stratégies mixtes sur les politiques Q_{-i}^t des autres agents l'agent i doit-il considérer, sachant que $(T-t)$ actions jointes ont été exécutées et que $(T-t)$ observations jointes ont été perçues auparavant ? Nous allons montrer que chaque historique d'actions et d'observations jointes possibles nous mène effectivement à un ensemble d'états de croyance probables.

Supposons que la politique jointe pour les $(T-t)$ premiers pas d'exécution ait été δ^{T-t}. Considérons maintenant un agent i, son point de vue sur le système, et plus précisément son information sur un autre agent j. Notons h_i^{T-t} le *fil d'exécution* de l'agent i, c'est-à-dire une historique d'actions et d'observations de longueur $(T-t)$. La connaissance de la politique jointe δ^{T-t}, des fonctions $\mathcal{T}$ et $\mathcal{O}$, et la formule de Bayes permettent alors de déterminer la probabilité $P(h_j^{T-t}|h_i^{T-t}, \delta^{T-t})$ de tout fil d'exécution h_j^{T-t}.

L'agent j doit terminer son exécution avec une des politiques utiles $q_j^t \in Q_j^t$, déjà déterminées par l'algorithme. La probabilité d'un fil d'exécution revient donc à la probabilité d'une politique, et c'est ainsi que l'agent i peut déterminer les croyances possibles sur les politiques de l'agent j. En générant des fils d'exécution h_{-i}^{T-t} pour tous les agents, il est également possible de calculer une distribution sur les états sous-jacents du système, et c'est ainsi que l'agent i peut déterminer les croyances possibles sur les états.

En somme, tout agent i est capable de construire un ensemble d'états de croyance pertinents en (a) générant une politique jointe δ^{T-t} pour les $(T-t)$ premiers pas d'exécution, (b) choisissant un fil d'exécution local h_i^{T-t}, (c) déterminant les pro-

babilités des fils d'exécution h_{-i}^{T-t} des autres agents et (d) construisant un ensemble d'états de croyance consistant avec δ^{T-t}, h_i^{T-t} et les politiques possibles des autres agents Q_{-i}^t. Cet algorithme est résumé en figure 3.

Entrée : les ensembles de politiques $Q_1^{t-1}, \ldots, Q_n^{t-1}$ à horizon (t-1).

1) Générer **toutes** les politiques jointes possibles δ^{T-t}.

2) Pour tout agent i :

a) Pour tout fil d'exécution h_i^{T-t} possible :

i) Créer $\overline{Q}_i^t$, l'ensemble exhaustif des politiques candidates, à partir de Q_i^{t-1}

ii) Générer l'ensemble de **tous** les états de croyance possibles $B(h_i^{T-t}, \delta^{T-t}, Q_{-i}^{t-1})$.

iii) Pour tout $b_i^t \in B(h_i^{T-t}, \delta^{T-t}, Q_{-i}^{t-1})$:

$$Q_i^t \leftarrow Q_i^t \bigcup \left\{ \arg\max_{q_i^t \in \overline{Q}_i^t} V(b_i^t, q_i^t) \right\}$$

b) Remplacer $\overline{Q}_i^t \leftarrow Q_i^t$.

Sortie : les ensembles de politiques $Q_1^t, \ldots, Q_n^t$ à horizon t.

Figure 3. *L'opérateur de programmation dynamique multi-agent à base de points*

Nous appelons par la suite programmation dynamique à base de points *exacte* l'approche qui répète les parties de (a) à (d) pour toutes les configurations possibles. Comme elle ne considère que les états de croyance possibles, elle génère en général moins de politiques que l'approche de [HAN 04]. Néanmoins, elle doit considérer un nombre exponentiel d'états de croyance. C'est pour cela que nous introduisons dans la partie suivante une version *approximative* de notre approche.

Theorème 3.1. *L'algorithme de programmation dynamique multi-agent à base de points exacte produit un ensemble de politiques utiles pour chaque agent.*

Démonstration. Voir [SZE 06]. □

3.3. *Programmation dynamique multi-agent à base de points approchée*

L'approximation de l'algorithme de programmation dynamique à base de points évite la génération exhaustive de politiques et d'états de croyance exigée par l'algorithme de la figure 3. Au lieu de considérer **toutes** les politiques jointes possibles comme indiqué dans la ligne 1) de l'algorithme, nous adoptons une stratégie déjà utilisée par [PIN 03], à savoir la génération aléatoire d'un ou de plusieurs représentants.

Nous pouvons utiliser la distance de Manhattan comme une métrique entre politiques, ce qui nous permet une meilleure répartition, en ne gardant que les politiques les plus éloignées les unes des autres. Nous choisissons par ailleurs d'éviter la génération de **tous** les états de croyance comme exigé par la ligne 2) a) ii). Chaque fil d'exécution possède en général une probabilité d'apparition différente, et nous pouvons établir une borne d'erreur ϵ sur le résultat final si nous évitons de considérer les fils d'exécution h_{-i}^{T-t} dont la probabilité d'apparition ne dépasse pas γ :

$$\epsilon \leq \frac{\gamma}{t(R_{max} - R_{min})} \tag{5}$$

4. Résultats expérimentaux

Nous avons implanté la version exacte et la version approchée de notre algorithme sur un problème du contrôle décentralisé déjà étudié dans le domaine. Nous précisons que la version approchée que nous considérons consiste à générer une seule politique jointe, mais de considérer tous les fils d'exécution possibles. Les valeurs de la version approchée sont alors moyennées sur 10 essais.

Le problème que nous considérons, introduit et spécifié dans [HAN 04], simule un réseau de communication simplifié. Les nœuds du réseau sont connectés entre eux, et pour chaque nœud i, une probabilité p_i d'arrivée d'un nouveau message est définie. Le message doit être diffusé à tous les nœuds voisins, mais une collision a lieu si deux nœuds essayent d'utiliser conjointement le même canal. Les messages ne sont alors pas envoyés et restent dans leurs tampons. L'objectif consiste à maximiser le nombre de messages transmis, et une récompense de $+1$ est donnée pour chaque message transmis.

L'obstacle majeur de l'algorithme de programmation dynamique multi-agent de [HAN 04] est l'explosion de sa consommation d'espace mémoire. Ainsi, l'algorithme ne peut pas résoudre le problème du réseau de communication pour des horizons plus grands que 4. La figure 4 montre le nombre de politiques utiles, et donc la consommation de l'espace mémoire, identifiés par l'approche de programmation dynamique de Hansen et al. et nos approches à base de points (exacte et approchée). Alors que l'approche classique doit garder jusqu'à 300 politiques en mémoire, la considération explicité des états de croyance possibles réduit le nombre de politiques utiles considérablement. Nous sommes ainsi capables de traiter des problèmes plus larges.

Puisque nous évitons la résolution d'un programme linéaire, nécessaire pour supprimer les politiques dominées, et puisque nous considérons moins de politiques que l'approche de Hansen et al., le temps de calcul de notre approche est également réduit. La figure 5 résume les temps de calcul pour le problème du réseau de communication. Il est intéressant de constater que le temps de calcul connaît un pic pour l'horizon 5 pour redescendre après, mais notons que chaque horizon représente a priori un autre problème, et donc nécessite de considérer un autre ensemble de points de croyance. Notons aussi que l'échelle est logarithmique.

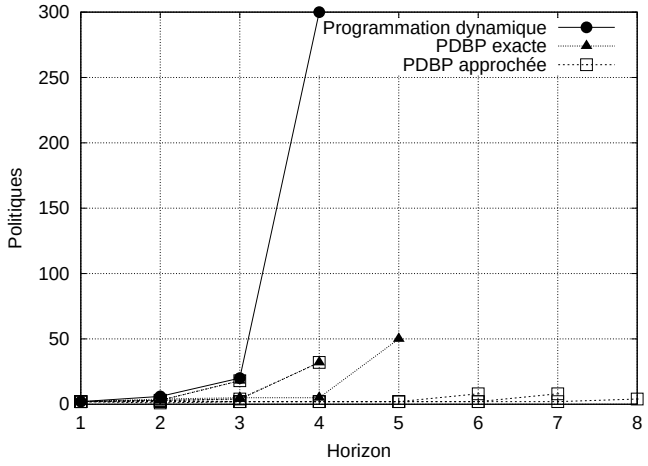

Figure 4. *Le nombre de politiques utiles pour le problème du réseau de communication, évaluées par l'algorithme de Hansen* et al.*, la version exacte et finalement la version approchée de l'approche de programmation dynamique à base de points (PDBP).*

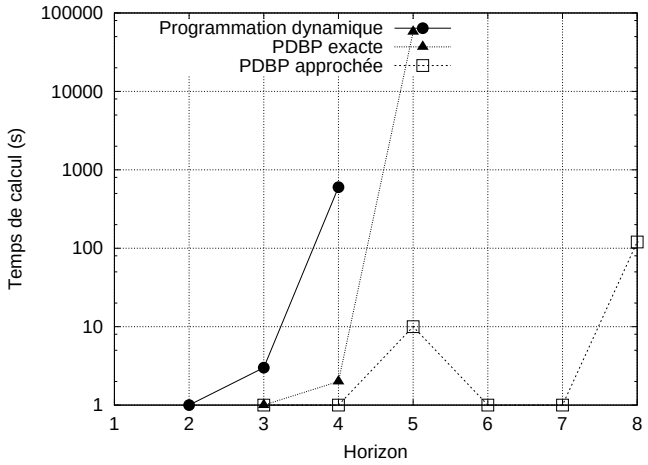

Figure 5. *Le temps de calcul pour les trois approches de programmation dynamique multi-agent sur le problème du réseau de communication.*

5. Conclusion et travaux futurs

Nous avons présenté un nouvel algorithme de programmation dynamique pour la résolution exacte ou approchée des DEC-POMDPs. Notre algorithme permet de ré-

soudre des problèmes à plus grande taille qu'auparavant en exploitant la structure particulière de l'espace des croyances multi-agents. Nous désignons maintenant quelques pistes de recherche afin de compléter ou étendre notre travail. Ainsi, il serait intéressant d'établir une borne d'erreur générale pour la version approchée de notre algorithme, comme c'est déjà le cas pour la version mono-agent de l'algorithme de programmation dynamique à base de points [PIN 03]. Nous sommes également confiants qu'il est possible d'appliquer la théorie des automates fini, déjà utilisée pour résoudre des DEC-POMDPs à horizon infini [BER 05, SZE 05a], à la programmation dynamique à base de points.

6. Bibliographie

[ALT 00] ALTMAN E., « Applications of Markov Decision Processes in Communication Networks : A Survey », rapport n° RR-3984, 2000, INRIA Sophia-Antipolis.

[BEC 04] BECKER R., ZILBERSTEIN S., LESSER V., GOLDMAN C. V., « Solving Transition Independent Decentralized Markov Decision Processes », *Journal of Artificial Intelligence Research*, vol. 22, 2004, p. 423-455.

[BER 02] BERNSTEIN D. S., GIVAN R., IMMERMAN N., ZILBERSTEIN S., « The Complexity of Decentralized Control of Markov Decision Processes », *Mathematics of Operations Research*, vol. 27, n° 4, 2002, p. 819-840.

[BER 05] BERNSTEIN D. S., HANSEN E. A., ZILBERSTEIN S., « Bounded Policy Iteration for Decentralized POMDPs », *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2005.

[CLA 98] CLAUS C., BOUTILIER C., « The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems », *AAAI/IAAI*, 1998, p. 746-752.

[EME 04] EMERY-MONTEMERLO R., GORDON G., SCHNEIDER J., THRUN S., « Approximate Solutions for Partially Observable Stochastic Games with Common Payoffs », *Proceedings of the 3rd AAMAS*, 2004.

[HAN 04] HANSEN E. A., BERNSTEIN D. S., ZILBERSTEIN S., « Dynamic Programming for Partially Observable Stochastic Games », *Proceedings of the 19th National Conference on Artificial Intelligence*, 2004.

[LOV 91] LOVEJOY W. S., « Computationally Feasible Bounds for Partially Observed Markov Decision Processes », *Operations Research*, vol. 39, n° 1, 1991, p. 162-175.

[NAI 03] NAIR R., TAMBE M., YOKOO M., PYNADATH D., MARSELLA S., « Taming Decentralized POMDPs : Towards Efficient Policy Computation for Multiagent Settings », *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.

[PES 00] PESHKIN L., KIM K.-E., MEULEAU N., KAELBLING L., « Learning to Cooperate via Policy Search », *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

[PIN 03] PINEAU J., GORDON G., THRUN S., « Point-based value iteration : An anytime algorithm for POMDPs », *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.

[SHI 05] SHIRAI Y., OSGOOD A. J., ZHAO Y., KELLY K. F., TOUR J. M., « Directional Control in Thermally Driven Single-Molecule Nanocars », *Nano Letters*, vol. 5, n° 11,

2005.

[SZE 05a] SZER D., CHARPILLET F., « An Optimal Best-first Search Algorithm for Solving Infinite Horizon DEC-POMDPs », *Proceedings of the 16th European Conference on Machine Learning*, 2005.

[SZE 05b] SZER D., CHARPILLET F., ZILBERSTEIN S., « MAA* : A Heuristic Search Algorithm for Solving Decentralized POMDPs », *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[SZE 06] SZER D., CHARPILLET F., « Point-based Dynamic Programming for DEC-POMDPs », *Proceedings of the 21st American National Conference on Artificial Intelligence*, 2006.