Une méthode de programmation linéaire mixte pour les POMDP décentralisé à horizon fini

Raghav Aras, Alain Dutech, François Charpillet

Une méthode de programmation linéaire mixte pour les POMDP décentralisé à horizon fini.

Raghav Aras, Alain Dutech, François Charpillet

MAIA - LORIA/INRIA
Campus Scientifique, BP239, 54506 Viller les Nancy
{aras,dutech,charp}@loria.fr
http://maia.loria.fr

Résumé : Nous nous intéressons au problème consistant à trouver une politique jointe optimale pour n agents dans le cadre du contrôle optimal d'un processus décisionnel de Markov décentralisé partiellement observé (Dec-POMDP). Le principe de notre approche est le suivant : la politique jointe optimale d'un Dec-POMDP est équivalente à une politique *sous-optimale* du POMDP lié, politique qui devrait en outre respecter des contraintes structurelles afin qu'elle puisse être décentralisée. En s'appuyant sur ce principe, nous présentons un algorithme exact qui utilise la programmation linéaire mixte (PLM) pour trouver un vecteur de poids de réalisation de séquences jointes (suite d'actions et d'observations jointes) qui représente ainsi une politique jointe. La politique jointe (décentralisable) optimale pour le Dec-POMDP dérive directement de la solution de ce PLM. Des expérimentation de notre algorithme sur des problèmes de Dec-POMDP standards montrent qu'il est plus efficace (rapide) que les algorithmes exacts actuels de programmation dynamique.

Mots-clés : Dec-POMDP, Programmation linéaire, Syst. multi-agents

1 Introduction

Le contrôle coopératif multi-agent d'un processus décisionnel de Markov (MDP) dans sa forme la plus générale peut se décrire comme un processus décisionnel de Markov décentralisé partiellement observé (Dec-POMDP) (Bernstein *et al.*, 2002). L'observabilité partielle signifie que les agents ne connaissent pas l'état exact du processus, état qui fait partie d'un ensemble S. A la place de la connaissance de cet état, les agents peuvent en général calculer un *belief-state*, c'est-à-dire une distribution de probabilité sur cet ensemble S. La décentralisation du processus implique entre autres que les *belief-states* des agents qui seront identiques au début du processus, évolueront de manière différente pour chaque agent individuel.

Chaque agent prend part au contrôle du Dec-POMDP pendant un certain nombre d'itérations (qu'on appelle l'*horizon*) par le biais d'une *politique*. Une politique est une fonction qui assigne une action à chaque *belief state*. Le profil formé des politiques de tous les agents forme une *politique jointe*. Une politique jointe optimale maximise la

récompense que les agents reçoivent à la fin de l'horizon. Décentralisation du contrôle et observabilité partielle font que la recherche d'une politique jointe optimale est d'une complexité prohibitive car le problème est NEXP-dur (Bernstein *et al.*, 2002). En pratique, les méthodes classiques essaient de gérer cette complexité en utilisant la *programmation dynamique* et plusieurs algorithmes exacts ont été proposés (Wilson (1972), Hansen *et al.* (2004), Emery-Montemerlo *et al.* (2004), Szer *et al.* (2005), Szer & Charpillet (2006)). Ces algorithmes sont cependant très gourmands en terme de mémoire et de temps de calcul.

Dans cet article, nous explorons une nouvelle approche qui se démarque de la programmation dynamique et qui, en pratique, s'avère plus efficace en terme de temps de calcul. L'idée principale de notre approche est qu'on peut exprimer la politique jointe d'un Dec-POMDP comme étant la politique d'un POMDP classique ((Sondik, 1971)), mois dur à résoudre (problème PSPACE-dur). Bien évidemment, en général la politique optimale π_p^* du POMDP ne coïncide pas avec la politique jointe optimale π_d^* du Dec-POMDP. Nous montrons que chercher à optimiser le POMDP en y ajoutant des contraintes structurelles **C** pour garantir la décentalisation des politiques permet de trouver une solution optimales du Dec-POMDP. De plus, nous proposons une *formulation séquentielle* du Dec-POMDP, ce qui permet de résoudre plus efficacement le POMDP associé en utilisant un *programme linéaire mixte* de notre cru, tenant justement compte des contraintes **C**. Nous développons plus formellement cette approche dans la suite de l'article et la confrontons à deux problèmes standards des Dec-POMDP.

Notre approche se situe dans la lignée des travaux de (Koller *et al.*, 1994), mais les améliorent de deux façons. D'une part, les algorithmes proposés par Koller ne sont applicables que pour 2 agents. D'autre part, (Koller *et al.*, 1994) s'appuient sur un programme linéaire complémentaire pour trouver une politique jointe qui est un équilibre de Nash, et donc potentiellement sous-optimal d'un point de vue global. Cette même limitation s'applique à la méthode de (Nair *et al.*, 2003; Chadès *et al.*, 2002) qui utilisent aussi une recherche d'équilibre. L'algorithme présenté dans (Koller & Megiddo, 1996) remédie à ce problème en cherchant *toutes* les politiques jointes formant un équilibre de Nash. Cependant c'est un gaspillage de ressources car dans un cadre coopératif (comme les Dec-POMDP), de nombreux équilibres sont sous-optimaux. Notre algorithme se repose sur le fait que les agents sont coopératifs pour trouver une politique optimale.

Le reste de cet article est organisé comme suit. Nous décrivons d'abord le cadre formel des Dec-POMDP et définissons une politique jointe optimale. Ensuite, nous présentons la formulation séquentielle d'un Dec-POMDP et la représentation d'une politique sous forme de poids de réalisation associés à des séquences. Vient ensuite la description du programme linéaire mixte qui utilise les poids de réalisation pour trouver une politique optimale jointe pour le Dec-POMDP. Puis nous donnons quelques améliorations immédiates de notre algorithme. Enfin, nous exposons les résultats des expérimentations effectuées avant de terminer sur une discussion suivie d'une conclusion.

2 Le cadre Dec-POMDP

Nous allons travailler avec des Dec-POMDP (Bernstein *et al.,* 2002) pour des ensemble $N = \{1, \ldots, n\}$ d'agents. Classiquement, ce sont des tuples **D** $(S, \{A_i\}_{i \in N}, \{\Omega_i\}_{i \in N}, \mathcal{T}, \mathcal{O}, \mathcal{R}, b_0, H)$, $\mathcal{T}$ encode les probabilité de transition entre états et $\mathcal{R}$ la fonction de récompense. b_0 est le *belief state* initial et que H est l'horizon du problème.

2.1 Politique et politique jointe

Chaque agent utilise une politique pour contrôler le Dec-POMDP **D**. Une politique d'horizon H peut se représenter par un arbre de profondeur H. Chaque noeud d'une politique d'un agent $i \in N$ contient une action de A_i et a autant de fils que d'observations possible dans Ω_i. Pour une politique π_i d'un agent i, $a(\pi_i)$ est l'action du noeud racine de π_i. Le sous-arbre associé à la branche o_i d'une politique π_i est noté $\pi_i(o_i)$. Une *politique jointe* d'horizon t est un tuple $\pi = (\pi_1, \ldots, \pi_n)$ où π_i est la politique d'horizon t de l'agent i. Etant donné une politique jointe π, $a(\pi)$ et $\pi(o)$ se déduisent naturellement des notations précédentes.

2.2 Politique jointe optimale

La récompense totale atteinte par une politique $\pi = (\pi_1, \ldots, \pi_n)$ d'horizon H appliqué au *belief state* b_0 est appelée la *valeur* de π et est notée $V(\pi, b_0)$. On l'obtient comme suit :

$$V(\pi, b_0) = \sum_{s \in S} b_0[s] V^H(\pi, s)$$

où V^H est la fonction valeur d'horizon H de **D**. La fonction valeur d'horizon t de **D** s'exprime récursivement par,

$$V^t(\pi, s) = R(s, a(\pi)) + \sum_{o \in \Omega} \sum_{s' \in S} \delta \times V^{t-1}(\pi(o), s')$$

où $\delta = \mathcal{T}(s, a(\pi), s')\mathcal{O}(a(\pi), s', o)$. V^H peut être calculée avec une induction arrière (Hansen *et al.,* 2004). Une politique jointe *optimale* est une politique jointe de valeur maximale. Les politiques individuelles d'une politique jointe optimale sont appelées des politiques optimales. De plus, nous pouvons parler sans ambiguïté de la valeur de **D**, qui est la valeur de la politique jointe optimale de **D**, que nous noterons **V**.

3 Représentation sous forme séquentielle

Intéressons-nous maintenant à la représentation sous forme séquentielle (Koller *et al.*, 1994) d'un POMDP **D**, qui est entièrement équivalente à la formulation classique. Pour cela nous aurons besoin des définitions suivantes :

Définition 1
Une séquence de longueur t *d'un agent* i *est une suite* $(a_i^1, o_i^1, \ldots, o_i^{t-1}, a_i^t)$ *où, pour chaque* $j \leq t$, $a_i^j \in A_i$ *et pour chaque* $j < t$, $o_i^j \in \Omega_i$.

Pour l'agent i, la séquence de longueur 0 est notée $\varnothing_i$. Un tuple $\alpha = (\alpha_1, \ldots, \alpha_n)$ où chaque α_i est une séquence de longueur t pour l'agent $i \in N$ est appelée une *séquence jointe* de longueur t. α peut aussi s'écrire comme (α_{-i}, α_i) où $\alpha_{-i} = (\alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n)$ est appelée une séquence jointe i*-réduite* de longueur t. Une séquence ou une séquence jointe de longueur H est dite *terminale*.

Définition 2
Un historique de longueur t *d'un agent* i *est une suite* $(a_i^1, o_i^1, \ldots, a_i^t, o_i^t)$ *où, pour chaque* $j \leq t$, $a_i^j \in A_i$ *et pour chaque* $j \leq t$, $o_i^j \in \Omega_i$.

Un tuple $(h_1, \ldots, h_n)$ où chaque h_i est un historique de longueur t pour un agent $i \in N$ est appelé un *historique joint* de longueur t.

Etant donné un historique h_i de longueur supérieure à 0, on note $\alpha_i(h_i)$ la séquence obtenue en enlevant la dernière observation de h_i. De même, étant donné un historique h_i et une action a_i, on note (h_i, a_i) la séquence obtenue en ajoutant a_i à la fin de h_i. La t-ème action et la t-ème observation de la séquence α_i sont respectivement notées $a(\alpha_i, t)$ et $o(\alpha_i, t)$. Ces notations s'appliquent de manière similaire aux séquences jointes et aux historiques joints.

Un Dec-POMDP **D** peut être décrit sous forme séquentielle par le tuple composé de :

- $\mathcal{H}_i$ est l'ensemble des historiques de longueurs comprise entre 0 et $H-1$ pour l'agent $i \in N$. $\mathcal{H} = \mathcal{H}_1 \times \ldots \times \mathcal{H}_n$ est l'ensemble des historique joints.
- $\mathcal{A}_i$ est l'ensemble des séquences de longueur comprise entre 0 et H pour l'agent $i \in N$. L'ensemble des séquences de longueur t pour l'agent i est notée $\mathcal{A}_i^t$. L'ensemble des séquences jointes de longueur t est noté $\mathcal{A}^t$. L'ensemble des séquences jointes i-réduites de longueur t est noté $\mathcal{A}_{-i}^t$.
- $R^{b_0} : \mathcal{A}^H \rightarrow \Re$ est la fonction récompense des séquences jointes terminales. $R^{b_0}(\alpha)$ est la récompense que les agents reçoivent si on *obtient* la séquence terminale α en partant du *belief state* initial b_0.

Les éléments N, $\mathcal{T}$ et $\mathcal{O}$ du tuple sont les mêmes que ceux de la définition originale de **D**.

3.1 Récompense des séquences jointes terminales

Nous expliquons maintenant comment calculer les récompenses liées aux séquences jointes terminales. Chaque séquence jointe terminale $\alpha = (\alpha_1, \ldots, \alpha_n)$ génère $H-1$ *belief states* consécutifs, en débutant en b_0. Le k-ème de ces *belief state* est noté $b(\alpha, k)$. Pour toutes les séquences, $b(\alpha, 0)$ est exactement b_0. La valeur de $b(\alpha, k)$ est calculée de la manière suivante :

$$b^{ao}(\alpha,t)[s] \leftarrow \sum_{s' \in S} \delta b(\alpha,t-1)[s'] \quad \text{avec } \delta = \mathcal{T}(s',a(\alpha,t),s)\mathcal{O}(a(\alpha,t),s,o(\alpha,t))$$

$$P(\alpha,t) \leftarrow \sum_{s \in S} b^{ao}(\alpha,t)[s] \quad \text{et} \quad b(\alpha,t)[s] \leftarrow \frac{b^{ao}(\alpha,t)[s]}{P(\alpha,t)}$$

La récompense $R^{b_0}(\alpha)$ pour α est la somme des récompenses pour chaque *belief state* rencontré le long de α multiplié par la probabilité d'obtenir α :

$$R^{b_0}(\alpha) = \prod_{t=1}^{H-1} P(\alpha,t) \sum_{t=1}^{H} \sum_{s \in S} b(\alpha,t-1)[s] \times R(s,a(\alpha,t))$$

Il est important de noter que cette récompense tient compte de la *probabilité d'occurrence* de la séquence.

3.2 Poids de réalisation des politiques

En utilisant la forme séquentielle, nous pouvons définir la politique π_i d'un agent i comme une fonction $\pi_i : \mathcal{H}_i \rightarrow A_i$. Pour une politique π_i on peut associer à chaque séquence un *poids de réalisation* qui vaut 1 *si et seulement si* la probabilité d'occurrence de cette séquence est non-nulle si on applique la politique π_i, sinon ce poids vaut 0. On dit qu'une séquence de poids non nul est dans le support de π_i. On peut ainsi associer à chaque politique π_i un vecteur w_{π_i} qui contient les poids de réalisation de toute séquence de $\mathcal{A}_i$. Le poids de réalisation d'une séquence α_i en appliquant π_i est noté $w_{\pi_i}(\alpha_i)$. Par convention, $\forall\ \pi_i$, $w_{\pi_i}(\varnothing_i) = 1$ (la séquence "nulle" est "présente" dans chaque politique). Un vecteur de poids de réalisation w_{π_i} sera appelé un rw-vecteur. Les poids de réalisation d'une séquence jointe (et donc les rw-vecteurs joints) sont définis de manière analogue. Il est à noter que les valeurs des poids de réalisation d'une séquence *ne dépendent pas de b_0 ou des politiques suivies par les autres agents*. Politique et rw-vecteur sont deux facette d'un même concept, et peuvent être inter-changés.

Les poids de réalisation doivent satisfaire certaines propriété.

– **Contrainte de politique**. Si une séquence est dans le support de π_i, alors une de ces successeurs est aussi dans le support. $\forall\ i \in N$, pour chaque séquence non-terminale α_i, $\forall\ o_i \in \Omega_i$,

$$w_{\pi_i}(\alpha_i) = \sum_{a_i \in A_i} w_{\pi_i}(\alpha_i, o_i, a_i) \tag{1}$$

De plus, pour être "décentralisable", une politique doit avoir certaines propriétés qui se reflètent au niveau des poids.

– **Contrainte de complétude**. Le nombre de séquences terminales dans chaque politique π_i vaut $T_i = |\Omega_i|^{H-1}$. Si on pose $T = \prod_{i \in N} T_i$,

$$\sum_{\alpha \in \mathcal{A}^H} w_\pi(\alpha) = T \tag{2}$$

– **Contrainte de décentralisation**. Comme les poids de réalisation valent soit 0, soit 1, une séquence jointe aura un poids de réalisation de 1 si et seulement si chacune de ses séquences a un poids de réalisation de 1. Etant donné $\alpha = (\alpha_1, \ldots, \alpha_n)$, cela s'exprime par,

$$w_\pi(\alpha) = \prod_{i \in N} w_{\pi_i}(\alpha_i) \tag{3}$$

On peut alors exprimer la valeur d'une politique jointe $\pi = (\pi_1, \ldots, \pi_n)$ au moyen de ses rw-vecteurs comme suit :

$$V(\pi, b_0) = \sum_{\alpha \in \mathcal{A}^H} \left[\prod_{i \in N} w_{\pi_i}(\alpha_i) \right] R^{b_0}(\alpha)$$

où α_i est la séquence terminale de l'agent i au sein de α. En fait, $\prod_{i \in N} w_{\pi_i}(\alpha_i)$ vaut 1 seulement toutes les séquences α_i sont possibles.

3.3 Poids de réalisation optimaux

Un vecteur de l'ensemble $\{0,1\}^{|\mathcal{A}_i|}$ et qui satisfait aux contraintes exprimées par l'équation (1) est appelé un rw-vecteur *valide* pour l'agent i (il représente bien une politique possible de l'agent). De même, tout vecteur w^p de l'ensemble $\{0,1\}^{|\mathcal{A}|}$ qui satisfait à l'équation (4) pour toute séquence jointe non terminale α et toute observation jointe o est appelé un rw-vecteur joint *valide* (il représente une politique jointe possible).

$$w^p(\alpha) = \sum_{a \in A} w^p(\alpha, o, a) \tag{4}$$

Soit W_i^d l'ensemble des rw-vecteurs valides de l'agent i, où $W^d = W_1^d \times \ldots \times W_n^d$. Alors, la valeur de la politique jointe optimale s'exprime par :

$$\max_{w^d \in W^d} V(w^d) = \max_{w^d \in W^d} \sum_{\alpha \in \mathcal{A}^H} \left[\prod_{i \in N} w_i^d(\alpha_i) \right] R^{b_0}(\alpha) \tag{5}$$

La politique optimale jointe peut ainsi être déterminée par le biais d'un programme non-linéaire dont la fonction objectif est donnée par l'équation (5) et dont les contraintes linéaires garantissent la propriété 1 (eq. (1)). Cependant, dans ce cas, la fonction $V(w^d)$ n'est pas forcément convexe. L'optimisation de fonctions non convexes est un problème NP-dur. Nous préférons donc exprimer la valeur de la politique jointe optimale en utilisant une fonction linéaire convexe des rw-vecteurs joints valides. Si W^p est l'ensemble de ces rw-vecteurs joints valides, nous pouvons réécrire l'équation (5) sous la forme :

$$\max_{w^p \in W^p} V(w^p) = \max_{w^p \in W^p} \sum_{\alpha \in \mathcal{A}^H} w^p(\alpha) R^{b_0}(\alpha) \tag{6}$$

Les équations (1)-(3) et (6) décrivent les 4 propriétés qui forment la base de notre formulation de problème avec un PLM pour trouver une politique jointe optimale.

4 Programmation Linéaire Mixte

4.1 Formulation

Le problème de programmation linéaire mixte (PLM) **M** qui permet de trouver la politique jointe optimale est décrit à la Figure 1. **M** identifie un rw-vecteur joint valide (le vecteur w^p) et un ensemble de rw-vecteurs valides (les vecteurs w^d). Les contraintes du PLM représentent les trois propriétés (équations (1) à (3)) que doit respecter toute politique jointe à travers ses rw-vecteurs. La fonction objectif capture la quatrième propriété (équation 6) de la politique optimale jointe.

On peut noter que **M** sans les contraintes (4.b) et (4.c) est un programme linéaire **L** qui permet d'optimiser le POMDP **P** lié au Dec-POMDP **D**. La fonction objectif de **L** est l'équation (6) qui est identique à la fonction objectif de **M** dont les contraintes (équation (4)) sont celles qui contraignent la politique dans **M**. **L** permet d'identifier un rw-vecteur joint qui maximise **V**. Cependant, comme les variables w^p sont continues (elles prennent leur valeur dans $[0, 1]$), le rw-vecteur joint peut ne pas être valide (il faudrait que les valeurs soient 0 ou 1). C'est pourquoi nous avons introduit les variables w^d et les contraintes (4.b) et (4.c) dans **L** pour assurer que le rw-vecteur joint soit valide (mais sous optimal pour **P**). Ainsi, les contraintes qui imposent des valeurs entières aux variables w^d associées des séquences terminales et les contraintes (4.b) et (4.c) assurent la validité de w^p. Les valeurs entières associées aux séquences terminales sont propagées vers les séquences non-terminales à travers les *contraintes de politiques*. L'équation 3 s'exprime à travers la contrainte (4.c) qui exige que chaque séquence terminale apparaissent dans exactement $T_{-i} = \frac{T}{T_i}$ séquences jointes terminales. Ceci assure que la politique jointe sera "décentralisable".

Ainsi, nous pouvons dire que **M** optimise en fait **P** *avec les contraintes* **C**. L'ensemble des politiques de longueur H pour **P** contient l'ensemble Γ^H des politiques jointes de longueur H de **D**. Autrement dit, si chaque politique de **P** n'est pas forcément une politique jointe valide, toutes les politiques jointes sont des politiques de **P**. Les contraintes **C** restreignent les solutions de **P** aux éléments de Γ^H. Les rw-vecteurs $w^d = (w_1^d, \ldots, w_n^d)$ de la solution optimale de **M** représentent une politique jointe optimale de **D**. Plus précisément, c'est la politique jointe $\pi^d = (\pi_1^d, \ldots, \pi_n^d)$ où chaque π_i^d est donné par : $\forall\ h_i \in \mathcal{H}_i$,

$$\pi_i^d(h_i) = \arg\max_{a_i \in A_i} w_i^d(h_i, a_i). \tag{7}$$

4.2 Amélioration

Trouver une solution optimale à un PLM est généralement un problème NP-dur. Pour essayer de diminuer cette complexité, en pratique, nous suggérons deux modifications au PLM **M** : l'élagage des séquences dominées (algorithme noté $\mathbf{M}^+$) et l'ajout de bornes à la fonction objectif (algorithmes $\mathbf{M}_{low}$ et $\mathbf{M}_{up}$) de manière à guider la résolution du problème. Des combinaisons de ces améliorations sont bien sûr possible.

1. variables :
   (a) variables continues
      i. $\forall\, \alpha \in \mathcal{A}^H$, $w^p(\alpha)$.
      ii. $\forall i \in N, \forall\, t < H, \forall \alpha_i \in \mathcal{A}_i^t$, $w_i^d(\alpha_i)$.
   (b) variables entières. $\forall i \in N, \forall \alpha_i \in \mathcal{A}_i^H$, $w_i^d(\alpha_i)$.
2. limites : (a) $\forall i \in N$, $w_i^d(\varnothing_i) = 1$, (b) $0 \leq w^p \leq 1$, (c) $0 \leq w^d \leq 1$.
3. fonction objectif à maximiser : $\mathbf{V} = \sum_{\alpha \in \mathcal{A}^H} w^p(\alpha) R^{b_0}(\alpha)$
4. avec les contraintes **C** suivantes :
   (a) contraintes de politique $\forall i \in N, \forall h_i \in \mathcal{H}_i$,

$$w_i^d(\alpha_i(h_i)) - \sum_{a_i \in A_i} w_i^d(h_i, a_i) = 0$$

   (b) contraintes de complétude de la politique jointe

$$\sum_{\alpha \in \mathcal{A}^H} w^p(\alpha) - T = 0$$

   (c) contraintes de décentralisation $\forall i \in N, \forall\, \alpha_i \in \mathcal{A}_i^H$

$$T_{-i} w_i^d(\alpha_i) - \sum_{\alpha_{-i} \in \mathcal{A}_{-i}^H} w^p(\alpha_{-i}, \alpha_i) = 0$$

FIG. 1 – Le programme linéaire mixte **M** qui permet de trouver une politique optimale jointe pour un Dec-POMDP.

4.2.1 Elagage des séquences dominées

Le principe de base utilisé pour transformer un problème de programmation linéaire en un PLM est celui du *branch and bound* (Fletcher, 1987) : on cherche la solution optimale parmi un *arbre* de programmes linéaires n'ayant que des variables continues et des variables entières *fixées*. Le facteur de branchement dépend du nombre de variables entières du PLM, il est donc intéressant de minimiser le nombre de variables entières, avec des répercussions évidentes au niveau du temps de calcul.

Le nombre de variables entières de **M** peut être réduit en élaguant les séquences *dominées* des ensembles $\mathcal{A}_i$. La notion de séquence dominée est similaire à la notion de politique dominée (Hansen *et al.*, 2004). Définissons d'abord les séquences *terminales* dominées. Pour cela, nous dirons qu'une séquence β_i est *observation-équivalente* à une autre séquence α_i si elle est de même taille k que α_i et que sa séquence d'observation est identique à celle de α_i : $\forall\, t < k$, $o(\alpha_i, t) = o(\beta_i, t)$. On appelle alors $E(\alpha_i)$ l'ensemble des séquences qui sont *observation*-équivalentes à α_i (sauf α_i elle-même).

Définition 3
Une séquence terminale α_i *est* dominée *s'il existe une mesure de probabilité* p *sur* $E(\alpha_i)$ *telle que* $\forall \alpha_{-i} \in \mathcal{A}^H_{-i}$,

$$R^{b_0}(\alpha_{-i}, \alpha_i) \leq \sum_{\beta_i \in E(\alpha_i)} p[\beta_i] R^{b_0}(\alpha_{-i}, \beta_i) \tag{8}$$

Toute séquence terminale dominée peut être élaguée en suivant la procédure classique de l'*élagage itératif*. Une séquence non-terminale α_i est dite dominée si toutes les séquences du type (α_i, o_i, a_i) sont dominées. Elles peuvent aussi être éliminées.

Un historique h_i peut être éliminé de $\mathcal{H}_i$ si toutes les séquences (h_i, a_i) sont élaguées. Dès lors, il se peut qu'il n'y ait plus exactement T séquences jointes terminales ayant un poids de réalisation valant 1 dans la politiques optimale. De plus, une séquence peut ne pas apparaître dans exactement T_{-i} séquence jointes terminales ayant un poids de réalisation valant 1. Nous prenons ces faits en compte en modifiant le PLM **M** en un nouveau PLM $\mathbf{M}^+$ dans lequel des variables w^d et w^p ne sont associés qu'à des séquences (jointes ou non-jointes) qui ne sont pas dominées. Les contraintes (4.b) et (4.c) sont elles aussi modifiées :

4 (b) :

$$\sum_{\alpha \in \mathcal{A}^H} w^p(\alpha) - T \leq 0$$

4 (c) : $\forall i \in N, \forall\, \alpha_i \in \mathcal{A}^H_i$,

$$T_{-i} w^d_i(\alpha_i) - \sum_{\alpha_{-i} \in \mathcal{A}^H_{-i}} w^p(\alpha_{-i}, \alpha_i) \geq 0$$

Les "contraintes de politique" de $\mathbf{M}^+$ sont identiques à celles de **M**. Normalement, il y a moins de variables entières dans $\mathbf{M}^+$ que dans **M**. L'élagage des séquences préserve la validité des solutions car aucune séquence dominée ne peut avoir un poids de réalisation de 1 dans une politique optimale. Les variables de $\mathbf{M}^+$ sont notées w^{d+} et la politique jointe qui en est dérivée en utilisant l'équation (7) est notée π^{d+}.

4.2.2 Ajouter des bornes

Un solveur de PLM explore un arbre de PLM en utilisant le *branch and bound* (Fletcher, 1987). Le choix du chemin d'exploration de l'arbre joue un rôle important au niveau du temps de calcul. Par exemple, la solution peut se trouver à la fin du chemin. Et même si la solution est au début du chemin, il se peut que le solveur doive parcourir tout l'arbre pour déterminer que cette solution est bien optimale. Il est possible d'écourter cette exploration en utilisant des bornes : le solveur ne cherche plus une solution optimale, mais une solution dont la valeur est comprise dans des bornes données. Trouver des bornes étroites est en soi un problème ardu, mais dans nos travaux nous avons utilisé des bornes simples à obtenir. Par exemple, si on connaît une politique jointe optimale de taille t, une borne inférieur pour la politique optimale jointe de taille $t+1$ est :

$$V_{low} = V(\pi^*, b_0) + \max_{a \in A} \min_{s \in S} R(s, a)$$

Pour calculer une borne supérieur, on peut passer par la valeur **P**, le POMDP associé à **D**. On note cette valeur V_{up} (équation 6), on peut l'obtenir en résolvant le programme linéaire **L**. Des bornes plus étroites peuvent être calculées par des techniques de relaxation du Lagrangien. Ainsi, on ajoute une borne inférieure **V** - $V_{low} \geq 0$ et une borne supérieure V_{up} - **V** ≥ 0 aux contraintes de la fonctions objectif. **M** et $\mathbf{M}^{+}$ avec une borne inférieure sont respectivement notés $\mathbf{M}_{low}$ et $\mathbf{M}^{+}_{low}$; respectivement $\mathbf{M}_{up}$ et $\mathbf{M}^{+}_{up}$ quand on y ajoute une borne supérieure.

5 Expérimentations

Nous avons testé nos algorithmes sur deux benchmarks de Dec-POMDP à 2 agents : le problème du broadcast sur un canal de communication à accès multiple (MABC) (Hansen *et al.*, 2004) et le problème du tigre multi-agent (MA-tiger) (Nair *et al.*, 2003) (dont nous ne rappelons pas les formulations). Ces problèmes représentent deux types de problèmes de contrôle coopératifs multi-agents car, pour le MABC la politique optimale π^p du POMDP correspondant au Dec-POMDP est la même que la politique optimale jointe π^d du Dec-POMDP alors que pour le MA-tiger, π^p est largement sous-optimale dans le POMDP associé. C'est dû à la nature déterministe du MABC, et cela se traduit par le fait que 75% des séquences on pu être élaguées dans le cas du MABC alors que *aucune* séquence n'a pu être écartée dans le cas du MA-tiger.

Nous avons utilisé le logiciel commercial ILOG Cplex 10 pour résoudre les PLM. Nous avons comparé nos algorithmes avec trois algorithmes exacts existants : DP (Hansen *et al.*, 2004), MAA* (Szer *et al.*, 2005) et PBDP (Szer & Charpillet, 2006). Une sélection de ces résultats sont rassemblé dans le tableau 1. Ces résultats sont encourageants car notre algorithme est significativement plus performant que les autres algorithmes exacts. Les temps d'exécutions précis de l'algorithme PBDP ne sont pas connus, mais nous savons que MAA* est plus performant que PBDP pour ce problème (Szer & Charpillet, 2006). Les résultats montrent que les bornes jouent un rôle très important dans les performances de notre algorithme. Notons que, même sans ces bornes, notre algorithme a des temps d'exécution plus courts que ceux de DP et MAA*, alors que ce dernier s'appuie aussi sur une méthode de recherche avant qui utilise des bornes.

6 Discussion

Nous avons présenté un algorithme exact qui s'appuie sur la formulation en séquences pour résoudre des Dec-POMDP à horizon fini. Des expériences sur deux problèmes de benchmark ont montré que cet algorithme est plus efficace que les autres algorithmes exacts existants. Des extensions de ce travail sont possibles dans plusieurs directions.

En premier lieu, les contraintes **C** du programme linéaire mixte **M** garantissent la décentralisation de la politique du POMDP. Nous pensons qu'il est possible d'utiliser un ensemble de contraintes différent pour néanmoins offrir les mêmes garanties. Autrement dit, un portefeuille de contraintes, adaptées à différents types de problèmes, pourraient être développé. Deuxièmement, les techniques de *plans de coupe* pourraient être incorporées à l'approche générale de manière à ne plus avoir besoin des contraintes

MABC (Hansen *et al.*, 2004)

H	**V**	**M**	$\mathbf{M}_{low}$	$\mathbf{M}_{up}$	$\mathbf{M}^{+}$	$\mathbf{M}^{+}_{low}$	$\mathbf{M}^{+}_{up}$	DP	MAA*	PBDP
3	2.99	0.86	0.93	1.03	0.84	0.84	0.93	5	t_3	1.0
4	3.89	900	900	907	80	120	10.2	$\approx 10^3$	t_4	2.0
5	4.79	-	-	-	$*$	$*$	25	-	-	$\approx 10^5$

MA-tiger, première version (Nair *et al.*, 2003)

H	**V**	**M**	$\mathbf{M}_{low}$	$\mathbf{M}_{up}$	$\mathbf{M}^{+}$	$\mathbf{M}^{+}_{low}$	$\mathbf{M}^{+}_{up}$	DP	MAA*	PBDP
3	5.19	3.7	4.9	3.5	6.4	7.6	6.2	?	t_3	$\approx t_3$
4	4.80	$*$	72	$*$	$*$	175	$*$	?	t_4	$\approx t_4$

MA-tiger, deuxième version Nair *et al.* (2003)

H	**V**	**M**	$\mathbf{M}_{low}$	$\mathbf{M}_{up}$	$\mathbf{M}^{+}$	$\mathbf{M}^{+}_{low}$	$\mathbf{M}^{+}_{up}$	DP	MAA*	PBDP
3	30	0.95	1.0	1.6	3.6	3.7	4.3	?	t_3	$\approx t_3$
4	40	$*$	43	$*$	$*$	146	$*$	?	t_4	$\approx t_4$

TAB. 1 – Nous avons indiqué les temps d'exécutions en secondes pour nos algorithmes (cela inclut l'élagage des séquences dominées et le calcul des bornes) et pour trois autres algorithmes exacts. Le problème du MA-tiger a été testé avec deux fonctions de récompense différentes. t_3 représente "plusieurs secondes" et t_4 représente "plusieurs heures" (Szer *et al.*, 2005). Nous avons noté '$-$' les cas où notre algorithme déclenchait un dépassement de mémoire. '$*$' indique qu'un time-out de 30 minutes a été atteint. '?' indique que nous n'avons pas d'indication de temps.

assurant que certaines variables sont entières. Le principe de cette approche est d'ajouter progressivement des contraintes jusqu'à obtenir une solution valide. Enfin, plus généralement, l'approche consistant à utiliser une formulation séquentielle pourrait être utilisée sur des sous-classes plus simples des Dec-POMDP, comme par exemple les POMDP et les MDP décentralisés.

Alors que la "programmation mathématique" a déjà été utilisée pour résoudre des problèmes mono-agent comme les MDP (Puterman, 1994), cette approche n'a été que peu étudiée dans le cadre des problèmes coopératifs multi-agents comme les Dec-POMDP. Les deux exceptions notables sont : (Amato *et al.*, 2006) qui décrit un programme non-linéaire dont la solution permet de trouver une politique jointe optimale de *taille fixée* pour des Dec-POMDP à horizon fini ; (Petrik & Zilberstein, 2007) présente une méthode PLM pour résoudre des Dec-MDP en horizon infini où on cherche à optimiser la récompense moyenne. Nous pensons que notre algorithme est une contribution intéressante à ce domaine.

Références

AMATO C., BERNSTEIN D. & ZILBERSTEIN S. (2006). Optimal fixed-size controllers for decentralized pomdps. *In Proceedings of the Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*.

BERNSTEIN D., GIVAN R., IMMERMAN N. & ZILBERSTEIN S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, **27(4)**, 819–840.

CHADÈS I., SCHERRER B. & CHARPILLET F. (2002). A heuristic approach for solving decentralized-pomdp : Assessment on the pursuit problem. *In Proceedings of the Sixteenth ACM Symposium on Applied Computing*.

EMERY-MONTEMERLO R., GORDON G., SCHNEIDER J. & THRUN S. (2004). Approximate solutions for paritally observable stochastic games with common payoffs. *In Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

FLETCHER R. (1987). *Practical Methods of Optimization*. New York : John Wiley & Sons.

HANSEN E., BERNSTEIN D. & ZILBERSTEIN S. (2004). Dynamic programming for partially observable stochastic games. *Proceedings of the 19th National Conference on Artificial Intelligence*, p. 709–715.

KOLLER D. & MEGIDDO N. (1996). Finding mixed strategies with small supports in extensive form games. *International Journal of Game Theory*, **25(1)**, 73–92.

KOLLER D., MEGIDDO N. & VON STENGEL B. (1994). Fast algorithms for finding randomized strategies in game trees. *In Proceedings of the 26th ACM Symposium on Theory of Computing*, p. 750–759.

NAIR R., PYNADATH D., YOKOO M. & TAMBE M. (2003). Taming decentralized pomdps : Towards efficient policy computation for multiagent settings. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, p. 705–711.

PETRIK M. & ZILBERSTEIN S. (2007). Average-reward decentralized markov decision processes. *In Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

PUTERMAN M. L. (1994). *Markov Decision Processes*. New York : John Wiley & Sons.

SONDIK E. J. (1971). The optimal control of partially observable markov processes. *Ph.D. Thesis, Stanford University*.

SZER D. & CHARPILLET F. (2006). Point-based dynamic programming for dec-pomdps. *In Proceedings of the 21st National Conference on Artificial Intelligence*.

SZER D., CHARPILLET F. & ZILBERSTEIN S. (2005). Maa* : A heuristic search algorithm for solving decentralized pomdps. *In Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*.

WILSON R. (1972). Computing equilibria of two-person games from the extensive form. *Management Science*, **18**, 448–460.