Des PDMs distribués interactifs pour le problème de l'allocation de tâches

Hosam HANNA Abdel-Illah Mouaddib
hanna@info.unicaen.fr mouaddib@info.unicaen.fr

GREYC - Université de Caen
6 Bd Maréchal Juin
14032 Caen

Résumé :
Le problème de l'allocation de tâches dans les environnements stochastiques peut être formalisé en un processus décisionnel de Markov (PDM). La construction et la résolution d'un tel PDM, de façon centralisée (PDMC), exigent qu'un agent possède des informations précises sur le comportement incertain des autres agents. Prendre en compte cette incertitude entraîne la réalisation de très longs calculs, ce qui fait qu'un PDMC est inapplicable pour de larges systèmes. Dans cet article, nous considérons un système multi-agent coopératif décentralisé et nous re-formalisons le problème de l'allocation de tâches en PDM Distribué (PDMD). Ce dernier est dérivé du PDMC par la distribution de l'espace d'états et du calcul de la fonction de gain. Chaque agent développe son PDMD sans connaître le comportement des autres. L'interaction entre les agents permet de coordonner les PDMDs et d'obtenir une allocation des tâches sans conflits. Nous présentons une démonstration mathématique de l'équivalence entre l'allocation de tâches obtenue d'une façons centralisée et celle obtenue via les PDMDs coordonnés.

Mots-clés : Décision sous incertitude, Décision distribuée, Processus décisionnel de Markov

Abstract:
The task allocation problem in the stochastic environments can be formalized by a Markov Decision Process (MDP). The construction and the resolution of such a MDP, by a centralized way (MDPC), require that an agent have precise information about the uncertain behavior of the other agents. Taking into account this uncertainty involves the realization of long calculations, and make the PDMC inapplicable for large systems. In this article, we consider a decentralized cooperative multi-agent system and we reformalize the task allocation problem by distributed MDPs (DMDP). This last is derived from the CMDP by the distribution of the states space and the calculation of the function of gain. Each agent develops its DMDP without having knowledge about the the others. The interaction between agents allows to coordinate the DMDPs and to obtain a task allocation with no conflicts equivalent to the task allocation obtained via a CMDP.

Keywords: Decision-making under uncertainty, Distributed decision, Markov Decision Process

1 Introduction

Le problème de l'allocation de tâches peut être caractérisé par deux points principaux. D'une part, un ensemble d'agents ayant des ressources limitées, et d'autre part, la répartition et l'exécution d'un ensemble de tâches par ces agents de façon à maximiser le gain du système. Ce problème a été largement abordé dans différents contextes : système centralisé [12, 13], système décentralisé [1] ou par la formation des coalitions [14]. Cependant, les méthodes proposées dans ces approches ignorent le comportement incertain de l'agent et l'effet de cette incertitude sur le gain du système. Elles considèrent que la consommation des ressources par un agent lors de l'exécution d'une tâche est connue à l'avance. Cette hypothèse, qui est très forte, rend ces méthodes inapplicable dans le cadre des environnements stochastiques, où la consommation des ressources est incertaine. Nous entendons par consommation incertaine des ressources (exécution incertaine des tâches) l'incapacité de l'agent à déterminer la quantité de ressources qui sera consommée lors de l'exécution d'une tâche. À cause de cette incertitude, un agent ne peut être sûr de pouvoir exécuter toutes les tâches qui lui sont allouées. Ainsi, le gain réel du système ne peut être déterminé qu'après l'exécution de toutes les tâches. Cependant, ce gain peut être pré-évalué (anticipé) au cours de la phase de l'allocation ; nous parlons donc de *gain espéré*. Ce dernier définit ce que les agents (le système) espèrent gagner en allouant une tâche à un agent. Le but des agents devient alors d'allouer les tâches de façon à maximiser le gain espéré du système. Dans un système centralisé, une allocation optimale des tâches peut être obtenue via un Processus Décisionnel de Markov centralisé (PDMC) [4]. L'inconvénient de cette solution est que le PDMC est construit par un seul agent, appelé «contrôleur», qui doit avoir une parfaite connaissance du comportement incertain des autres agents. L'espace d'états induit par ce PDMC est très large, ce qui rend sa résolution coûteuse pour ne pas dire infaisable. Plusieurs approches ont proposé des méthodes pour réduire le coût de la résolution des PDMs à vaste espace d'états [5, 10, 9]. Mais, les solutions obtenues sont approximatives et l'efficacité de ces méthodes dépend de la possibilité de décomposer l'espace d'états en plusieurs régions à faible connexion. Dans notre contexte, les agents agissent d'une façon dis-

tribuée. Ils ont pour but de trouver une allocation des tâches similaire à celle obtenue par un PDMC. Afin d'atteindre cet objectif, nous distribuons l'espace d'états et le calcul à travers les agents via des PDMs *Distribués* (PDMD). En effet, chaque agent développe son propre PDMD sans connaître le comportement incertain des autres. Afin d'éviter les conflits (l'allocation d'une même tâche à plusieurs agents), les agents coordonnent leur PDMD en communiquant et échangeant quelques valeurs. Contrairement aux PDMs partiellement observables, dans lesquels un agent prend sa décision en se fondant sur une observation incomplète des autres, l'interaction entre les différents agents permet à chacun de prendre ces décisions en fonction des autres. Ainsi, les PDMDs ne représentent pas un PDM décentralisé partiellement observable dont la résolution est un problème classé *NEXP-Difficile* [3].

Dans les deux sections suivantes, nous présentons notre cadre de travail ainsi que les PDMC et l'allocation optimale des tâches (sections 2 et 3). Ce qui nous permet par la suite (section 4, de détailler le PDMD et de montrer l'équivalence entre le PDMC et les PDMDs coordonnés. Enfin, nous discutons de la complexité de notre modèle dans la section 5.

2 Le cadre du travail

Notre système consiste en un ensemble A d'agents dont les ressources sont limitées et un ensemble T de tâches, où $A = \{a_1, a_2, \cdots, a_m\}$ et $T = \{t_1, t_2, \cdots, t_n\}$. Une allocation $\mathcal{P}$ des tâches est une répartition de l'ensemble T, telle que : $\mathcal{P} = \{B^1, B^2, \cdots, B^m\}$ où B^k est l'ensemble de tâches allouées à a_k. Les ensembles $B^k, k = 1, ..., m$ vérifient : $\forall k \neq l, B^k \cap B^l = \emptyset$ et $\bigcup_{k=1}^{m} B^k \subseteq T$. Nous supposons que les tâches doivent être allouées et exécutées dans un certain ordre comme par exemple : $t_1, t_2, ..., t_n$. Pour chaque tâche réalisée, le système reçoit un gain qui varie en fonction de l'agent exécutant cette tâche.

Définition 2.1 *Chaque agent a_k a une fonction du gain $g_k : T \rightarrow \mathtt{R}^+$, où $g_k(t_i)$ est le gain que le système obtient si a_k exécute t_i.*

L'allocation de t_i à a_k est une condition nécessaire mais pas suffisante pour obtenir le gain $g_k(t_i)$. Cela est dû à l'incertitude sur l'exécution des tâches (la consommation des ressources) qui rend l'agent incapable de déterminer a priori la quantité de ressources qui sera consommée lors de l'exécution de chaque tâche. Un agent a_k ne peut donc savoir s'il va pouvoir exécuter toutes les tâches qui lui sont allouées, ou s'il va être obliger d'en ignorer quelques-unes. Afin de formaliser l'exécution incertaine des tâches, nous utilisons une représentation discrète de la consommation des ressources.

Définition 2.2 *Chaque agent a_k a une représentation discrète finie de la consommation des ressources, tel que : l'exécution d'une tâche t_i par a_k consomme une des p quantités de ressources $r_i^1, r_i^2, \cdots, r_i^p$. La valeur de p peut varier en fonction de l'agent et de la tâche.*

Nous définissons ainsi la distribution de probabilité suivante :

Définition 2.3 *Chaque agent a_k a une distribution de probabilité : $PE_{a_k} : T \times \{r_i^j : t_i \in T, j = 1, ..., p\} \longrightarrow]0, 1]$, où $PE_{a_k}(t_i, r_i^j)$ est la probabilité que l'exécution de t_i par a_k consomme la quantité r_i^j.*

Les différentes quantités de ressources $r_i^j, i = 1, ..., n, j = 1, ..., p$ et la distribution de probabilité PE_{a_k} représentent la connaissance de l'agent a_k sur l'exécution incertaine des tâches. Ainsi, ces informations sont inaccessibles par les autres agents. Enfin, le but des agents est d'allouer les tâches de façon à maximiser le gain espéré du système. Par la suite, nous considérons un système centralisé et nous formalisons le problème de l'allocation de tâches en un PDMC.

3 L'allocation de tâches via PDMC

L'allocation de tâches peut être vu comme un processus décisionnel séquentiel. Dans chaque étape de ce processus, une décision, concernant l'allocation d'une tâche à un agent, est prise. L'étape suivante est consacrée à l'allocation de la tâche suivante. Nous définissons l'état du système par l'ensemble des tâches alloués à chaque agent et par les ressources disponibles. L'allocation d'une tâche change bien-entendu l'état du système. La décision d'allouer une tâche à un agent ne dépend que de l'état actuel du système ce qui fait de ce processus un processus markovien. Dans un système centralisé, un agent «contrôleur» construit et résout un PDMC et obtient une allocation optimale des tâches. Comme un PDM, un PDMC consiste en un ensemble d'états $\mathcal{S}$, un ensemble d'actions $\mathcal{AC}$ et un modèle de transition probabiliste. À chaque état

est associé un gain représentant ce que le système gagne dans cet état. Nous associons aussi, à chaque action, un gain espéré qui représente ce que le système espère gagner en appliquant cette action. Par la suite, nous détaillons le PDMC[1] correspondant à notre problème via : (1) les états, (2) les actions, (3) le modèle de transition et (4) le gain espéré.

3.1 La représentation de l'état

Un état de $\mathcal{S}$ représente une situation de l'*allocation des tâches* et de l'*anticipation* de la consommation des ressources. Nous notons par $S_i = ((B_i^1, R_i^1), ..., (B_i^m, R_i^m))$ l'état du système à l'étape i, où :

– B_i^k est l'ensemble de tâches allouées à a_k jusqu'à l'étape i. L'ensemble B_i^k vérifie les conditions suivantes :

1. $\bigcup_{k=1}^{m} B_i^k \subseteq \{t_1, ..., t_i\}$
2. $\forall k, \forall l,$ if $k \neq l$ alors $B_i^k \cap B_i^l = \emptyset$

Une tâche $t_{j \leq i}$ appartient à B_i^k si a_k possédait assez de ressources pour l'exécuter au moment où la décision d'allouer t_j à a_k a été prise (voir la section suivante) ;

– R_i^k sont les ressources disponibles de l'agent a_k

La construction du PDMC commence de l'état initial $S_0 = ((\emptyset, R_{a_1}), ..., (\emptyset, R_{a_m}))$, où R_{a_k} sont les ressources initiales dont l'agent a_k dispose. Le système arrive à un état terminal quand toutes les tâches sont allouées ou lorsque les agents ne disposent plus de ressources. Un état terminal est noté par :
$S_n = ((B_n^1, R_n^1), ..., (B_n^m, R_n^m))$, où $\bigcup_{k}^{m} B_n^k \subseteq T$ et $R_n^k \geq 0, k = 1, \cdots, m$.

3.2 Les actions et le modèle de transition

Dans notre contexte, nous notons par $AL(t_i, a_k)$ une action de $\mathcal{AC}$, où $t_i \in T$ et $a_k \in A$. Elle consiste en l'allocation de la tâche t_i à l'agent a_k et en l'anticipation de la quantité de ressources qui sera consommée lors de l'exécution de cette tâche. Soit $S_{i-1} = ((B_{i-1}^1, R_{i-1}^1), ..., (B_{i-1}^m, R_{i-1}^m))$ l'état courant du système, l'application de l'action $AL(t_i, a_k)$ conduit le système à un des p états définis par :

$S_i^j = ((B_i^1, R_i^1), ..., (B_i^k, R_i^k = R_{i-1}^k - r_i^j), ..., (B_i^m, R_i^m))$, où $j = 1, \cdots, p$, et :

– $B_i^l = B_{i-1}^l$ et $R_i^l = R_{i-1}^l, \forall l \neq k$

– $B_i^k = \begin{cases} B_{i-1}^k \cup \{t_i\}, & \text{si } r_i^j \leq R_{i-1}^k \\ B_{i-1}^k, & \text{sinon} \end{cases}$

– $R_i^k = \begin{cases} R_{i-1}^k - r_i^j, & \text{si } r_i^j \leq R_{i-1}^k \\ 0, & \text{sinon} \end{cases}$

En effet, il y a p états possibles car l'exécution de t_i peut consommer une des p quantités de ressources. Dans le cas où $r_i^j > R_{i-1}^k$, l'agent va consommer toutes ses ressources (R_{i-1}^k) sans achever t_i. Cette tâche n'appartient donc pas à B_i^k. Cependant, l'arrivée à l'état S_i^j, où $r_i^j \leq R_{i-1}$, signifie que t_i a été exécutée et le gain $W_{a_k}(t_i)$ est obtenu. Par ailleurs, la probabilité de transition de l'action $AL(t_i, a_k)$ s'exprime par la distribution PE_{a_k} car a_k arrive à l'état S_i^j si l'exécution de t_i consomme la quantité r_i^j. Par conséquent, la probabilité d'obtenir le gain $W_{a_k}(t_i)$ est : $\sum PE_{a_k}(t_i, r_i^j)$ où $r_i^j \leq R_{i-1}$ et $j = 1, \cdots, p$.

3.3 Le gain espéré et la politique optimale

La décision d'appliquer une action dépend du gain que le système espère gagner en appliquant cette action. Nous notons par $Q(AL(t_i, a_k))$ le gain espéré associé à l'action $AL(t_i, a_k)$. Étant dans l'état $S_{i-1}, i = 1, \cdots, n$, une politique $\pi(S_{i-1})$ à suivre est une action $\{AL(t_i, a_k), a_k \in A\}$ à appliquer. Le gain espéré d'une politique $\pi(S_{i-1}) = AL(t_i, a_k)$ est simplement $Q(AL(t_i, a_k))$. Nous définissons le gain $V[S_{i-1}]$ associé à l'état S_{i-1} comme étant le gain immédiat α_{i-1} que le système obtient lorsqu'il est dans cet état, augmenté du gain espéré de la politique $\pi_{S_{i-1}}$ suivie (reward-to-go). Soit a_k l'agent auquel la tâche t_{i-1} a été allouée, le gain immédiat α_{i-1} est défini comme suit :

$$\alpha_{i-1} = W_{a_k}(t_{i-1}) \text{ if } t_{i-1} \in B_{i-1}^k, \ 0 \text{ sinon}$$

Une politique optimale $\pi^*(S_{i-1})$ est celle qui maximise le gain espéré dans chaque état. Nous pouvons formaliser $V[S_{i-1}]$ et $Q(Al(t_i, a_k))$ grâce aux équations de Bellman [2], qui peuvent être résolues par la valeur d'itération [11] :

[1] Le PDMC décrit ici est fondé sur le PDM présenté dans [4]

➤ pour chaque non-terminal état S_{i-1} :

$$V[S_{i-1}] = \alpha_{i-1} + \max_{a_k \in A}\{Q(AL(t_i, a_k))\} \quad (1)$$

$$Q(AL(t_i, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_i, r_i^j) \times V[S_i^j] \quad (2)$$

➤ pour les états terminaux S_n :

$$V[S_n] = \alpha_n \quad (3)$$

Le fait que le PDMC obtenu soit un horizon finit sans boucle[2], la politique dérivée de l'équation (1) est donc optimale [2]. D'une façon formelle,

$$\pi^*(S_{i-1}) = arg(\max_{a_k \in A}\{Q(AL(t_i, a_k))\}) \quad (4)$$

Cette équation représente la décision optimale du système dans un état S_{i-1} où $i = 1, \cdots, n$. L'opérateur arg retourne l'action dont le gain espéré est maximal. Dans le cas spécial où deux agents a_l et $a_h, l < h$ vérifient que $Q(AL(t_i, a_l)) = Q(AL(t_i, a_h)) = \max_{a_k \in A}\{Q(AL(t_i, a_k))\}$, l'opérateur arg retourne, par convention, $AL(t_i, a_l)$. Enfin, une allocation optimale des tâches $\mathcal{P}$ peut être obtenue par l'état terminal $S_n = ((B_n^1, R_n^1), ..., (B_n^m, R_n^m))$ atteint par l'application de la politique optimale dans chaque état à partit de S_0. Formellement, $\mathcal{P} = \{B_n^1, B_n^2, \cdots, B_n^m\}$. Par ailleurs, la construction et la résolution d'un PDMC est une opération coûteuse. En effet, cette complexité est due au large espace d'états[3] $|\mathcal{S}| = \frac{(m \times p)^{n+1} - 1}{(m \times p) - 1}$ et au fait que le calcul se fait par un seul agent.

Le problème dans les systèmes décentralisés est que la décision globale doit être prise par tous les agents afin d'éviter les conflits. En d'autres termes, si la politique optimale dans un état S_{i-1} est d'allouer la tâche t_i à l'agent a_k, alors chaque agent $a_{l \neq k}$ doit prendre localement la décision d'ignorer t_i et l'agent a_k doit prendre, localement, la décision de l'exécuter. Afin de permettre aux agents, agissant d'une façon distribuée, de prendre des décisions locales cohérentes et d'obtenir une allocation optimale des tâches, nous reformalisons le problème de l'allocation de tâches en PDMDs. Chaque agent construit un PDMD dans lequel l'allocation d'une tâche à lui-même est représentée de la même façon que dans le PDMC, cependant l'allocation de cette tâche à un autre agent est représentée par un seul état (au lieu de p états dans le PDMC). Grâce à cette représentation, l'agent réduit l'espace d'états considéré dans son PDMD comparativement avec celui du PDMC.

Dans ce qui suit, nous détaillons le PDMD et montrons comment les agents coordonnent leurs actions. Nous présentons ensuite un preuve mathématique de l'équivalence entre l'allocation optimale des tâches obtenue par les PDMDs et celle obtenue par le PDMC.

4 L'allocation de tâches via PDMDs

Tout comme un PDMC, un PDMD consiste en un ensemble d'états $\bar{\mathcal{S}}$, un ensemble d'actions $\bar{\mathcal{AC}}$ et un modèle de transition. Les différences principales entre un PDMC et PDMD sont : le modèle de transition et la manière de calculer le gain espéré. Nous décrivons par la suite le PDMD construit par un agent a_k et le calcul distribué réalisé.

4.1 La représentation d'état

Un état de $\bar{\mathcal{S}}$ représente une situation de l'allocation de tâches et de l'anticipation de la consommation des ressources de l'`agent` a_k. Nous notons par $\bar{S}_i = (B_i^k, R_i^k)$ l'état de l'agent a_k à l'étape i, où B_i^k et R_i^k ont les mêmes significations que dans le PDMC. La construction du PDMD commence à partir de l'état initial $\bar{S}_0 = (\emptyset, R_{a_k})$. À l'étape n, l'agent arrive à un état terminal $\bar{S}_n = (B_n^k, R_n^k)$, où $B_n^k \subseteq T, R_n^k \geq 0$.

4.2 Les actions et le modèle de transition

Nous distinguons dans l'ensemble $\bar{\mathcal{AC}}$ deux types d'actions, à savoir : les actions représentant l'allocation de tâches à l'agent a_k et les actions représentant l'allocation de tâches aux autres agents. L'action qui consiste à allouer une tâche t_i à l'agent a_k est notée ici par $al(t_i, a_k)$. Cette dernière a les mêmes significations que l'action $AL(t_i, a_k)$ (section (3.2)). Soit $\bar{S}_{i-1} = (B_{i-1}^k, R_{i-1}^k)$ l'état courant de l'agent a_k. L'application de l'action $al(t_i, a_k)$ dans l'état $\bar{S}_{i-1}$ conduit l'agent à un des p états suivants :

[2] Dans un état S_{i-1}, les actions applicables sont $AL(t_i, a_k), a_k \in A$. Nous remarquons qu'aucune d'elles ne peut conduire à un état $S_{j \leq i-1}$.

[3] En l'occurrence, l'heuristique proposée dans [8] est efficace pour résoudre les large PDM ayant des boucles, ce qui n'est pas le cas dans notre PDMC.

$\bar{S_i^j} = (B_i^k, R_i^k = R_{i-1}^k - r_i^j), j = 1, ..., p$, où :

- $B_i^k = \begin{cases} B_{i-1}^k \cup \{t_i\}, & \text{si } r_i^j \leq R_{i-1}^k \\ B_{i-1}^k, & \text{sinon} \end{cases}$

- $R_i^k = \begin{cases} R_{i-1}^k - r_i^j, & \text{si } r_i^j \leq R_{i-1}^k \\ 0 \quad , & \text{sinon} \end{cases}$

La probabilité de cette transition est aussi exprimée par la distribution PE_{a_k}. Quant à l'allocation de t_i, à un autre agent, elle n'est pas observée par l'agent a_k dans son PDMD. Par contre, cette allocation y est prise en compte grâce à une nouvelle action, à savoir l'action Ig. Cette dernière représente en effet la décision de l'agent a_k d'ignorer la tâche t_i. L'application de l'action $Ig(t_i, a_k)$ dans l'état $\bar{S}_{i-1}$ conduit l'agent a_k à l'état : $\bar{S}_i = (B_i^k = B_{i-1}^k, R_i^k = R_{i-1}^k)$. La probabilité de la transition engendrée par cette action est de 1 car l'agent ne consomme aucune ressources.

Par exemple, considérons trois agents a_1, a_2 et a_3 dont une partie de leur PDMD est représentée dans la figure (1). Nous supposons que $p = 2$ pour tous les agents et nous représentons les ressources par des valeurs entières. La figure (1) illustre comment chaque agent observe la transition correspondant à l'allocation de deux tâches t_i et t_{i+1}. Pour simplifier la notation, les états dans cette figure portent des numéros 1,...,39. L'allocation de la tâche t_i à a_1 conduit, dans son PDMD à un des deux états 2 et 3, cependant cette allocation conduit dans le PDMD de l'agent a_2 (*resp.* a_3) à l'état 17 (*resp.* 30). Soient $\bar{S}_{i-1}(B_{i-1}^1, 100)$ et $\bar{S}_{i-1}(B_{i-1}^2, 120)$ les états des agents a_1 et d'a_2 à l'étape $i-1$. Supposons que l'exécution de t_i par a_1 puisse consommer une des deux quantités de ressources $r_i^1 = 60$ et $r_i^2 = 105$. D'une façon formelle, l'état 2 est $(B_i^1 = B_{i-1}^1 \cup \{t_i\}, 40)$ et l'état 3 est $(B_i^1 = B_{i-1}^1, R_i^1 = 0)$. Dans le PDMD de a_2, l'état 17 est $(B_i^2 = B_{i-1}^2, R_i^2 = 120)$.

4.3 Le gain espéré et la politique optimale

Nous définissons le gain $V[\bar{S}_{i-1}]$ et le gain espéré $Q(al(t_i, a_k))$ de la même manière qu'étudiée dans la section (3.3). La différence ici est que l'agent a_k ne peut pas calculer le gain espéré $Q(al(t_i, a_{l \neq k})), \forall a_l \in A$. En fait, il va recevoir ces valeurs des agents correspondants. Le gain immédiat $\bar{\alpha}_{i-1}$ associé à l'état $\bar{S}_{i-1}$ est défini comme suit :

$$\bar{\alpha}_{i-1} = W_{a_k}(t_{i-1}) \text{ si } t_{i-1} \in B_{i-1}^k, \ 0 \text{ sinon}$$

L'agent a_k calcule, dans son PDMD, le gain $V[\bar{S}_{i-1}]$ de la façon suivante :

➤ pour chaque état non terminal $\bar{S}_{i-1}$:

$$V[\bar{S}_{i-1}] = \bar{\alpha}_{i-1} \ + \\ + \max\{Q(al(t_i, a_k)), Q(Ig(t_i, a_k))\} \quad (5)$$

$$Q(al(t_i, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_i, r_i^j) \times V[\bar{S}_i^j] \quad (6)$$

$$Q(Ig(t_i, a_k)) = \max_{a_{l \neq k} \in \mathcal{A}} \{Q(al(t_i, a_l))\} \quad (7)$$

➤ pour tout état terminal $\bar{S}_n = (B_n^k, R_n^k)$:

$$V[\bar{S}_n] \ = \bar{\alpha}_n \quad (8)$$

Une politique à suivre par l'agent est une action al ou Ig à appliquer. Une politique optimale est celle qui maximise le gain espéré à chaque état. Pour les mêmes raisons considérées dans la section (3.3), l'équation suivante :

$$\pi_k^*(\bar{S}_{i-1}) = arg(\max\{Q(al(t_i, a_k)), Q(Ig(t_i, a_k))\}) \quad (9)$$

formalise la politique optimale. Elle représente la décision optimale de l'agent a_k dans l'état $\bar{S}_{i-1}, i = 1, \cdots, n$. Dans cette équation, l'opérateur arg a la même signification que dans l'équation (4). Par ailleurs, l'équation (5) nécessite du calcul par l'agent a_k ($\bar{\alpha}_{i-1}$ et $Q(al(t_i, a_k))$) et du calcul par les autres agents (équation (7)). Cette dernière équation génère des communications entre les agents[4], ce qui va conduire aux décisions cohérentes comme nous allons voire par la suite. Afin d'obtenir une politique optimale, les agents échangent les valeurs $Q(al(t_i, a_l)), a_l \in A$; cela nécessite l'échange des $Q(al(t_{i+1}, a_l)), a_l \in A$, et ainsi de suite. Cependant, chaque agent a_l peut directement calculer $Q(al(t_n, a_l))$ grâce à l'équation (8). Après avoir échangé ces valeurs, les valeurs $V[\bar{S}_{n-1}]$ peuvent être calculées pour tout état $\bar{S}_{n-1}$. Connaître les valeurs $V[\bar{S}_{n-1}]$ permet de

[4]Des approches utilisent l'algorithme du graphe de coordination comme un outil de communication [7, 6]. Cet algorithme peut être efficace si un agent n'a pas besoin de se coordonner avec tous les autres (graphe incomplet). Dans notre cas, l'opérateur $\max$ dans l'équation (5) génère des communications avec tous les autres agents.

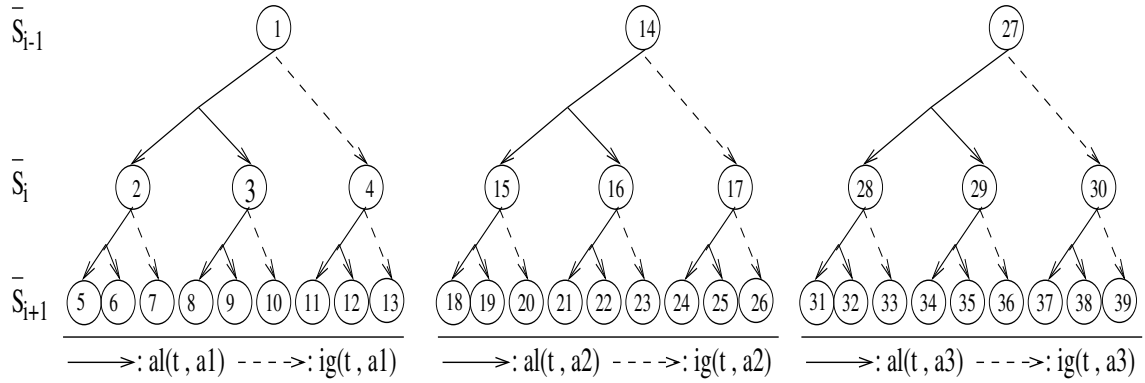

FIG. 1 – Le modèle de transition correspondant à l'allocation de deux tâches, dans les PDMDs des agents a_1, a_2 et a_3. Pour simplifier la notation, les états sont numérotés par 1,...,39.

calculer $V[\bar{S}_{n-2}]$, et ceci jusqu'à l'état $\bar{S}_{i-1}$. Par exemple, supposons que les agents a_1, a_2 et a_3 soient dans un état $\bar{S}_{i-1=n-2}$ (figure 1). Les états $\bar{S}_{i+1}$ sont donc les états terminaux $\bar{S}_n$. Afin d'obtenir la politique optimale, les agents calculent et échangent les valeurs suivantes :

- Les agents calculent, chacun dans son PDMD, les valeurs $V[\bar{S}_n]$, où tout $\bar{S}_n$ est un état engendré par une action $al(t_{n-1}, a_l), l = 1, ..., 3$ (les états 5, 6, 8, 9, 11 et 12 pour l'agent a_1 par exemple) ;

- Les agents calculent et échangent ensuite les valeurs $Q(al(t_n, a_{1,2,3}))$ où les actions $al(t_n, a_{1,2,3})$ sont appliquées dans les états $\bar{S}_{n-1}$ qui sont engendrés par des actions $Ig(t_{n_1}, a_{1,2,3})$, (l'état 4, 17 et 30 dans le PDMD de a_1, a_2 et a_3 respectivement) ;

- Dans le PDMD de a_1, la valeur $Q(Ig(t_n, a_1))$, où l'action Ig est appliquée dans les états 2 et 3, n'est que le maximum des deux valeurs $Q(al(t_n, a_2))$ et $Q(al(t_n, a_3))$ reçues. Les autre agents a_2 et a_3 calculent $Q(Ig(t_n, a_{2,3}))$, chacun dans son PDMD, de la même façon ;

- L'agent a_1 peut maintenant calculer les gains $V[\bar{S}_{n-1}]$ des états 3 et 4 (selon l'équation (5)) et par conséquent la valeur $Q(al(t_{n-1}), a_1)$. D'une façon similaire, les deux agents a_2 et a_3 calculent les valeurs $Q(al(t_{n-1}), a_2)$ et $Q(al(t_{n-1}), a_3)$, respectivement ;

- Les agents échangent les valeurs $Q(al(t_{n-1}, a_{\{1,2,3\}}))$.

Enfin, chaque agent applique l'équation (9) afin d'obtenir la politique optimale à suivre dans son PDMD. Les politiques optimales obtenues par les différents PDMDs ne créent pas de conflits entre les agents, comme nous pouvons le constater grâce au lemme suivant :

Lemme 4.1 *Dans chaque état* $\bar{S}_{i-1}, i = 1, ..., n$*, les politiques optimales obtenues par les agents selon l'équation (9) sont consistantes. Formellement,*
$\forall a_k \in A, i = 1, ..n, \pi_k^*(\bar{S}_{i-1}) = al(t_i, a_k) \Longrightarrow \pi_l^*(\bar{S}_{i-1}) = Ig(t_i, a_l), \forall a_l \neq a_k$.

Démonstration $\forall i = 1, ..., n$, nous avons :
$\forall a_k \in A, \pi_k^*(\bar{S}_{i-1}) = al(t_i, a_k) \stackrel{eq9}{\Longrightarrow}$
$al(t_i, a_k) = arg(\max\{Q(al(t_i, a_k)),$
$Q(Ig(t_i, a_k))\})$
$\stackrel{eq7}{\Longrightarrow}$
$al(t_i, a_k) = arg(\max\{Q(al(t_i, a_k)),$
$\max_{a_{l\neq k}}\{Q(al(t_i, a_l))\}\})$
$\Longrightarrow$
$Q(al(t_i, a_k)) > Q(al(t_i, a_l)), \forall a_{l\neq k} \in A$
Comme les valeurs $Q(al(t_i, a_h)), \forall a_h \in A$ sont identiques dans tous les PDMDs (l'agent a_h calcule $Q(al(t_i, a_h))$ et l'envoie aux autres), et vu la définition de l'opérateur arg alors l'inéquation précédente implique :

$\forall a_{l\neq k} \in A, al(t_i, a_l) \neq$
$arg(\max\{Q(al(t_i, a_l)), \max_{a_{h\neq l}}\{Q(al(t_i, a_h))\}\})$
car au moins $Q(al(t_i, a_k)) > Q(al(t_i, a_l))$.
Alors, $\forall a_{l\neq k} \in A, al(t_i, a_l) \neq$
$arg(\max\{Q(al(t_i, a_l)), Q(Ig(t_i, a_l))\})$
et donc $\forall a_{l\neq k} \in A, \pi_l^*(\bar{S}_{i-1}) = Ig(t_i, a_l)$ □

Nous démontrons par la suite l'équivalence entre le PDMC et les PDMDs coordonnés. Pour ce faire, nous présentons la relation entre l'espace d'états du PDMC et l'ensemble d'espaces d'états des PDMDs. Nous montrons ensuite l'égalité entre la politique optimale calculée dans le PDMC et celles calculées dans les PDMDs.

Lemme 4.2 *Pour chaque état* $S_i = ((B_i^1, R_i^1), ..., (B_i^m, R_i^m)), i = 0, ..., n$ *du PDMC, il existe un état* $\bar{S}_i$ *dans le PDMD de l'agent* $a_k, k = 1, ..., m$ *où* $\bar{S}_i = (B_i^k, R_i^k)$.

Démonstration pour $i = 0$, nous sommes à l'état initial $S_0 = ((\emptyset, R_{a_1}), ..., (\emptyset, R_{a_m}))$ dans le PDMD. Pour cet état, le couple $(\emptyset, R_{a_k}), k = 1, ..., m$ ce n'est que l'état initial dans le PDMD de l'agent a_k. Nous supposons que le lemme soit correcte pour tout état $S_{i-1} = ((B_{i-1}^1, R_{i-1}^1), ..., (B_{i-1}^m, R_{i-1}^m))$, nous montrons que il est encore correcte pour tout états S_i. En fait, les états S_i sont engendrés par l'application des actions $AL(t_i, a_k), a_k \in A$ dans les états S_{i-1}. Selon le modèle de transition (voir section 3.2), l'action $AL(t_i, a_k)$ n'a des effets que sur l'ensemble B_{i-1}^k et les ressources R_{i-1}^k. Donc, pour un état S_i engendré par cette action , tel que $S_i = ((B_i^1, R_i^1), ..., (B_i^m, R_i^m)$, il existe dans le PDMD de l'agent a_k un état $\bar{S}_i = (B_i^k, R_i^k)$ engendré par l'application de l'action $al(t_i, a_k)$ dans l'état (B_{i-1}^k, R_{i-1}^k), et il existe dans le PDMD d'un agent $a_{l \neq k} \in A$ un état $\bar{S}_i = (B_i^l, R_i^l)$ engendré par l'application de l'action $Ig(t_i, a_l)$. □

Lemme 4.3 *Pour chaque état* $S_{i-1} = ((B_{i-1}^1, R_{i-1}^1), ..., (B_{i-1}^k, R_{i-1}^k), ...(B_{i-1}^m, R_{i-1}^m))$ *du PDMC, le gain espéré associé à une action* $AL(t_i, a_k), \forall a_k \in \mathcal{A}$ *est égale à celui associé à l'action* $al(t_i, a_k)$ *appliquée dans l'état* $\bar{S}_{i-1}(B_{i-1}^k, R_{i-1}^k)$ *du PDMD de l'agent* a_k.

Démonstration Nous démontrons ce lemme par occurrence. Pour $i = n$, nous sommes dans un état $S_{n-1} = ((B_{n-1}^1, R_{n-1}^1), ..., (B_{n-1}^k, R_{n-1}^k), ...(B_{n-1}^m, R_{n-1}^m))$ dans le PDMC, et dans un état $\bar{S}_{n-1} = (B_{n-1}^k, R_{n-1}^k)$ dans le PDMD d'a_k. Selon les équations (2) et (3), nous avons :

$$Q(AL(t_n, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_n, r_n^j) \times V[S_n^j]$$

$$Q(AL(t_n, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_n, r_n^j) \times \alpha_n^j, \text{ où } \alpha_n^j$$

correspond à l'état S_n^j. Comme l'agent auquel la tâche t_n est allouée par l'action $AL(t_n, a_k)$ est a_k, alors $\alpha_n^j = \bar{\alpha}_n^j$, (définitions de α et $\bar{\alpha}$). Donc,

$$Q(AL(t_n, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_n, r_n^j) \times \bar{\alpha}_n^j$$

$$Q(AL(t_n, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_n, r_n^j) \times V[\bar{S}_n^j]$$

Selon l'équation (6), nous avons : $Q(AL(t_n, a_k)) = Q(al(t_n, a_k))$.
Supposons, maintenant, qu'à l'étape i le lemme est correcte, c-à-d :
$\forall S_i = ((B_i^1, R_i^1), ..., (B_i^k, R_i^k), ...(B_i^m, R_i^m))$, $Q(AL(t_{i+1}, a_k)) = Q(al(t_{i+1}, a_k)), \forall a_k \in A$ où al est appliquée dans l'état $\bar{S}_i(B_i^k, R_i^k)$. Nous prouvons par la suite que le lemme est encore correct pour $i - 1$.

$$Q(AL(t_i, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_i, r_i^j) \times V[S_i^j]$$

$$Q(AL(t_i, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_i, r_i^j) \times [\alpha_{\mathbf{i}}^{\mathbf{j}} + \max_{a_l \in A}\{Q(\mathbf{AL}(t_{i+1}, a_l))\}]$$

Selon lemme (1), nous avons :

$$Q(AL(t_i, a_k)) = \sum_{j=1}^{p} PE_{a_k}(t_i, r_i^j) \times [\bar{\alpha}_{\mathbf{i}}^{\mathbf{j}} + \max_{a_l \in A}\{Q(\mathbf{al}(t_{i+1}, a_l))\}]$$

$Q(AL(t_i, a_k)) = Q(al(t_i, a_k))$. □

Lemme 4.4 *Pour chaque état* $S_i = ((B_i^1, R_i^1), ..., (B_i^m, R_i^m))$ *du PDMC, la politique optimale* $\pi^*(S_i)$ *est équivalente aux politiques optimales* $\bar{\pi}_k^*(\bar{S}_i = (B_i^k, R_i^k)), a_k \in \mathcal{A}$ *obtenues via les PDMDs coordonnés. Formellement :*
$\forall a_k \in A, i = 1, ..., n, \pi_k^*(\bar{S}_{i-1}) = al(t_i, a_k) \Longleftrightarrow \pi^*(S_{i-1}) = AL(t_i, a_k)$

Démonstration Soit a_k l'agent vérifiant $\pi^*(\bar{S}_{i-1}) = al(t_i, a_k)$ à l'étape $i - 1$. Selon l'équation (9), nous avons :

$$al(t_i, a_k) = arg(\max\{Q(al(t_i, a_k)), Q(ig(t_i, a_k))\}) \overset{eq.7}{\Longleftrightarrow}$$

$$al(t_i, a_k) = arg(\max\{Q(al(t_i, a_k)), \max_{a_{l \neq k} \in \mathcal{A}}\{Q(al(t_i, a_l))\}\}) \Longleftrightarrow$$

$$Q(al(t_i, a_k)) > Q(al(t_i, a_{l \neq k})), a_l \in A \overset{lem.(4.3)}{\Longleftrightarrow}$$

$$Q(AL(t_i, a_k)) > Q(AL(t_i, a_{l \neq k})), a_l \in A \Longleftrightarrow$$

$$AL(t_i, a_k) = arg(\max_{a_l \in A}\{Q(AL(t_i, a_l))\}) \overset{eq.(4)}{\Longleftrightarrow}$$

$\pi^*(S_{i-1}) = AL(t_i, a_k)$ □

Une allocation optimale des tâches $\mathcal{P}$ peut être obtenue des états $\bar{S}_n = (B_n^k, R_n^k), a_k \in A$ atteints par l'application de la politique optimale dans chaque état à partir de l'état initial $\bar{S}_0^k$, formellement $\mathcal{P} = \{B_n^1, B_n^2, \cdots, B_n^m\}$.

5 Analyse et complexité

Dans cette article, nous avons décomposé l'espace d'états du PDMC et distribué le calcul de la fonction du gain à travers les agents. Cela facilite la résolution d'un PDMD comparativement au PDMC. En effet, dans un PDMD, il y a $\frac{(p+1)^{n+1}-1}{(p+1)-1}$ états, alors que l'espace d'états du PDMC contient $\frac{(m\times p)^{n+1}-1}{(m\times p)-1}$ états. Ainsi, la taille de l'espace d'états d'un PDMD est indépendante du nombre d'agents agissant dans le système, ce qui n'est pas le cas pour le PDMC. Cela permet d'augmenter le nombre d'agents dans le système sans complexifier le calcul de PDMDs. Quant aux messages échangés, un agent envoie $n \times (m-1)$ messages afin de déterminer une politique optimale étant dans l'état initial. Enfin, l'équivalence entre le PDMC et les PDMDs coordonnés, démontrée par le lemme (4.4), permet aux agents d'obtenir la même allocation optimale des tâches que celle obtenue par le PDMC.

6 Conclusion

Dans cet article, nous avons traité le problème de l'allocation de tâches dans un système multi-agent décentralisé caractérisé par des agents possédant une quantité limitée de ressources et par l'incertitude qui relève de l'exécution des tâches. Le PDMC utilisé pour des systèmes centralisés fournit une allocation optimale des tâches. Cependant, la résolution d'un tel PDM exige des longs calculs réalisés par un seul agent qui doit aussi avoir de parfaites connaissances sur le comportement incertain des autres. Le PDMD proposé dans cet article, est fondé sur la décomposition de l'espace d'états du PDMC et sur la distribution du calcul de la fonction du gain. Chaque agent construit son PDMD sans connaître le comportement incertain des autres. Il représente, dans son PDMD, l'allocation d'une tâches à tous les autres par un seul état. Cela réduit considérablement la taille de l'espace d'états des PDMDs et par conséquent la complexité du calcul à réaliser par chaque agent. L'interaction entre agents leur permet de coordonner les PDMDs et d'obtenir une allocation optimale des tâches similaire à celle obtenue par un PDMC.

Références

[1] Martin Andersson and Tuomas Sandholm. Time-quality tradeoffs in reallocative negotiation with combinatorial contract types. In *Proceedings of AAAI 99*, pages 3–10, jul 1999.

[2] R. E. Bellman. A markov decision process. *journal of Mathematical Mechanics*, pages 6 :679–684, 1957.

[3] S. Daniel Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. In *Uncertainty in Artificial Intelligence : Proceedings of the Sixteenth Conference (UAI-2000)*, pages 32–37, San Francisco, CA, 2000.

[4] Maroua Bouzid, Hosam Hanna, and Abdel-Illah Mouaddib. *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, chapter Controlled Social Deliberation : Deliberation Levels in Theoretic-Decision Approaches for Task Allocation in Resource-Bounded Agents, pages 198–216. LNAI 2103. Springer-Verlag Berlin Heidelberg, 2001.

[5] Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1121–1127, 1995.

[6] Carlos Gusetrin, Daphne Koller, and Parr Ronald. Multiagent planning with factored mdps. *In NIPS-14*, 2001.

[7] Carlos Gusetrin, Shobha Venkataraman, and Daphne Koller. Context-specific multiagent coordination and planning with factored mdps. *In the Eighteenth National Conference on Artificial Intelligence AAAI, Canada*, July 2002.

[8] Eric A. Hansen and Shlomo Zilberstien. Lao* : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, pages 35–62, 2001.

[9] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of markov decision processes using macro-actions. In *Uncertainty in Artificial Intelligence*, pages 220–229, 1998.

[10] Ronald Parr. Flexible decomposition algorithms for weakly coupled markov decision problems. In *Uncertainty in Artificial Intelligence*, pages 422–430, 1998.

[11] Martin L. Puterman. *Markov Decision Processes*. John Wiley & Sons, New York, 1994.

[12] M.H. Rothkopf, A. Pekec, and R.M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8) :1131–1147, 1998.

[13] Tuomas Sandholm. emediator : A next generation electronic commerce server. In *International Conference on Autonomous Agents (AGENTS), Barcelona, Spain*, june 2000.

[14] Onn Shehory and Sarit Kraus. Task allocation via coalition formation amoung autonomous agents. In *Proceedings of IJCAI 95*, pages 655–661, Montreal, 1995.