

Un processus décisionnel de Markov multi-agent pour la gestion de tâches

Hosam Hanna, Abdel-Ilhah Mouaddib

GREYC-Département d'Informatique
Université de Caen - Campus II
Bd du Maréchal Juin
14032 Caen Cedex, France
(hanna, mouaddib)@info.unicaen.fr

Dans les systèmes multi-agents et distribués, le problème de l'allocation des tâches suscite un intérêt de plus en plus important. En général, les techniques proposées ignorent *le coût de l'allocation et le comportement incertain de l'agent*. Dans cet article, nous traitons ces deux problèmes. Pour le premier, nous avons distribué l'allocation à travers les agents (afin de réduire le coût), ensuite nous avons coordonné les décisions locales des agents. Pour le deuxième problème, nous avons formalisé l'allocation et l'exécution des tâches par un *Processus Décisionnel de Markov* (PDM). En effet, le PDM permet aux agents de tenir compte de l'incertitude sur la consommation des ressources nécessaires à la réalisation de chaque tâche. L'autre point que nous avons abordé est de coordonner les décisions locales (choix locaux des tâches à réaliser) des agents. Pour cela, nous introduisons dans cet article deux stratégies de coordination fondées sur la distribution séquentielle de tâches.

Mots-clés : coordination, agents distribués, décision sous incertitude, processus décisionnel de Markov, agents coopératifs

1 INTRODUCTION

Le problème de l'allocation de tâches dans les systèmes multi-agents et distribués suscite un intérêt de plus en plus important. Ce problème est caractérisé par un ensemble d'agents à ressources limitées qui répartissent et exécutent un ensemble de tâches. De plus, les tâches doivent être réparties d'une façon à maximiser le gain du système. En effet, l'exécution d'une tâche représente un gain pour le système et varie en fonction de l'agent la

réalisant. Plusieurs travaux ont traité ce problème dans différents contextes [12, 13, 15, 14, 16, 17]. Bien que les solutions fournies par ces approches puissent être utiles pour différentes applications, le problème de trouver l'allocation optimale maximisant le gain total du système reste *NP*-difficile. Cette difficulté devient plus importante dans les environnements stochastiques où l'exécution des tâches est incertaine. Nous attendons par l'exécution incertaine d'une tâche, l'incapacité de l'agent de déterminer la quantité exacte des ressources à consommer lors de l'exécution de la tâche. Dans une telle situation, l'agent ne peut pas savoir à priori s'il va pouvoir exécuter toutes les tâches qui lui sont allouées ou s'il va être obligé d'en ignorer quelques unes. Par conséquent, le gain réel du système ne peut être déterminé qu'après la fin de l'exécution de toutes les tâches. Pour cette raison, l'allocation d'une tâche à un agent ne peut être vue comme un gain réel mais comme un *gain espéré*. Ce dernier représente en fait ce que le système espère gagner en allouant une tâche à un agent. L'allocation optimale des tâches est donc celle qui maximise le gain espéré du système. Quelques travaux ont étudié la maximisation du gain espéré dans les environnements stochastiques pour des systèmes non-coopératifs [3, 8] et centralisé coopératif [4]. Afin de faire face à l'incertitude, ces approches utilisent le *processus décisionnel de Markov* (PDM) qui permet de maximiser le gain espéré à long-terme. Dans un système centralisé coopératif, l'allocation optimale des tâches peut être obtenue grâce à un PDM global (PDMG) construit par un seul agent [4]. L'inconvénient principal de cette solution est la complexité du calcul nécessaire pour résoudre un PDMG, ce qui diminue l'utilité de l'utilisation d'un PDMG quand le nombre de tâches et d'agents est élevé. Des approches [6, 10, 9] ont proposé des méthodes pour réduire la complexité du calcul d'un PDM via la décomposition de l'espace d'états en plusieurs régions. Le désavantage de ces méthodes est la nécessité d'une partition de l'espace d'états dont le nombre de transitions entre régions est petit (un couplage faible entre régions). Cependant, la construction d'un PDMG engendre un espace d'états fortement lié.

Dans cet article, nous considérons un système décentralisé coopératif où les agents doivent se partager et exécuter un ensemble de tâches sous incertitude. Nous nous focalisons sur l'utilisation du PDM afin de faire face à l'incertitude. Bien évidemment, dans un système décentralisé la construction d'un PDMG par chaque agent est une opération coûteuse. En effet, en plus de la complexité du calcul, chaque agent doit avoir des connaissances précises sur le comportement des autres. Les stratégies que nous proposons, réalisent

la répartition des tâches en deux étapes, à savoir : (1) la sélection locale des tâches par chaque agent et (2) la coordination des choix locaux. Nous formalisons la sélection locale des tâches par un PDM Local (PDML). Cela permet à chaque agent d'obtenir un sous-ensemble de tâches qui maximise son gain espéré. Nous introduisons ensuite deux stratégies de coordination des choix locaux, appelées Tâche Par Tâche et Paquet Par Paquet, fondées sur l'allocation séquentielle des tâches. Dans les deux stratégies, chaque agent communique aux autres le gain qu'il peut obtenir s'il exécute les tâches de son choix optimal. Enfin, une tâche est allouée à l'agent ayant proposé un gain maximal. Il est important de rappeler que l'obtention d'une allocation optimale des tâches, dont le gain espéré est maximal, à partir des PDM locaux est un problème *NEXP-Difficile* [2].

On montre dans ce qui suit un exemple d'un système multi-agents dont l'exécution des tâches se fait sous incertitude. Nous présentons ensuite notre cadre de travail dans la section 3. Nous rappelons dans la section 4 comment construire le PDML correspondant au système centralisé. Dans la section 5, nous détaillons le développement de notre PDML. Les deux stratégies de coordination sont expliquées dans la section 6. La section 7 présente la complexité du PDML et la performance des stratégies. Nous discutons dans la section 8 les différentes approches concernant le problème de l'allocation des tâches. Enfin, nous concluons l'article dans la section 9.

2 EXEMPLE : ROBOTS PLANÉTAIRES

Il s'agit d'un ensemble de différents robots (agents) qui doivent se partager et réaliser un ensemble de tâches (prise des photos, découpage des roches, analyse chimique,...) sur une planète comme Mars [7]. Chaque agent dispose d'une batterie à capacité limitée (ressources) que l'exécution des tâches décharge. Le profit scientifique (le gain) obtenu à la fin de l'exécution d'une tâche varie en fonction de l'agent la réalisant. Par exemple, quand il s'agit de la prise d'une photo, la clarté et la résolution sont variées selon l'agent qui prend cette photo. Il est donc important d'allouer chaque tâche à l'agent pouvant fournir un gain maximal. Dans ce système, l'exécution des tâches est incertaine car les agents n'ont pas de connaissance précise de l'environnement. Considérons le découpage d'une roche et supposons que cette tâche soit allouée à un agent. Pour diverses raisons (l'existence d'un métal dur

dans la roche par exemple), l'exécution de cette tâche peut consommer une quantité d'énergie supérieure à ce qui a été prévu. Donc, il est possible que l'agent ne puisse pas exécuter toutes les autres tâches qui lui sont allouées. Par conséquent, il est nécessaire d'allouer les tâches aux agents en tenant compte de la consommation incertaine des ressources.

3 LE CADRE DU TRAVAIL

Nous considérons un ensemble de tâches $T = \{t_1, t_2, \dots, t_n\}$ et un ensemble d'agents $A = \{a_1, a_2, \dots, a_m\}$. Chaque agent a_k possède une quantité limitée R_{a_k} de ressources qu'il peut utiliser pour réaliser des tâches. L'exécution d'une tâche se traduit par un gain qui varie en fonction de l'agent qui exécute la tâche.

Définition 1

Chaque agent a_k possède une fonction de récompense g_{a_k} définie sur l'ensemble de tâches T :

$$g_{a_k} : T \longrightarrow R^+$$

telle que $g_{a_k}(t_i)$ est le gain que l'agent a_k obtiendra s'il exécute la tâche t_i .

Le gain total G_{a_k} d'un agent a_k à la fin de l'exécution des tâches est défini par :

$$G_{a_k} = \sum_{t_i \in B_{a_k}} g_{a_k}(t_i)$$

où B_{a_k} est l'ensemble de tâches exécutées par a_k . De même, nous définissons le gain total G du système à la fin de l'exécution des tâches :

$$G = \sum_{a_k \in A} G_{a_k}$$

Par ailleurs, l'incertitude sur l'exécution des tâches rend l'agent incapable de déterminer à priori la quantité de ressources qui sera consommée lors de l'exécution de chaque tâche. Afin de traiter cette incertitude, nous utilisons une représentation discrète de la consommation de ressources.

Définition 2

L'exécution d'une tâche t_i par un agent a_k consomme une des p quantités de ressources : $r_i^1, r_i^2, \dots, r_i^p$. Le nombre de quantités de ressources p varie en fonction de l'agent et de la tâche.

Nous définissons ainsi une distribution de probabilité qui représente la connaissance de l'agent sur l'exécution incertaine des tâches.

Définition 3

Chaque agent a_k a une distribution de probabilité :

$$PE_{a_k} : T \times \{r_i^j : t_i \in T, j = 1, \dots, p\} \longrightarrow]0, 1]$$

telle que : $PE_{a_k}(t_i, r_i^j)$ est la probabilité que l'exécution de t_i par a_k consomme la quantité r_i^j .

Par la suite, nous montrons comment répartir les tâches via un PDMG. Les inconvénients du PDMG dans les systèmes décentralisés sont ainsi présentés.

4 LE PROCESSUS DÉCISIONNEL DE MARKOV GLOBAL

Le PDMG consiste en un ensemble d'états \mathcal{S} , un ensemble d'action \mathcal{A} et en un modèle de transition [11]. Il est construit par un des agents, appelé central, afin d'allouer les tâches aux agents d'une façon maximisant le gain espéré du système. Nous décrivons dans la suite le PDMG par : (1) les états, (2) les actions, (3) le modèle de transition et (4) le gain espéré.

4.1 La représentation des états

Chaque état de \mathcal{S} représente une situation de l'allocation de tâches et de la consommation anticipée de ressources de l'ensemble des agents. Nous notons par $S_i = ((B_i^1, R_i^1), \dots, (B_i^m, R_i^m))$ l'état du système à l'instant i , tel que :

- B_i^k est l'ensemble des tâches allouées à l'agent a_k jusqu'à l'instant i . L'allocation d'une tâche à l'agent a_k se fait après l'anticipation de la quantité des ressources à consommer lors de l'exécution de cette tâche (Voir la section suivante). L'ensemble B_i^k vérifie :

$$B_i^k \subseteq \{t_1, \dots, t_i\}, B_i^k \cap B_i^l = \emptyset$$

- R_i^k sont les ressources disponibles de l'agent a_k à l'instant i .

La construction du PDMG commence à partir de l'état initial :

$$S_0 = ((\emptyset, R_{a_1}), \dots, (\emptyset, R_{a_m}))$$

À l'instant n , le système arrive à un état terminal :

$$S_n = ((B_n^1, R_n^1), \dots, (B_n^m, R_n^m))$$

qui représente la situation finale de l'allocation et de la consommation anticipée de ressources, c'est-à-dire : toutes les tâches sont allouées ou toutes les ressources sont supposées être consommées. La transition d'un état à un autre se fait par l'application d'une action.

4.2 Les actions et le modèle de transition

Dans notre contexte, une action, $AL(t_i, a_k) \in \mathcal{AC}$, consiste en l'allocation d'une tâche (t_i) à un agent (a_k) et en l'anticipation de la quantité de ressources qui sera consommée lors de l'exécution de cette tâche. L'application de cette action sur un état S_{i-1} , tel que :

$$S_{i-1} = ((B_{i-1}^1, R_{i-1}^1), \dots, (B_{i-1}^k, R_{i-1}^k), \dots, (B_{i-1}^m, R_{i-1}^m))$$

conduit le système à un des états suivants :

$$S_i = ((B_i^1, R_i^1), \dots, (B_i^k, R_i^k = R_{i-1}^k - r_i^j), \dots, (B_i^m, R_i^m))$$

où $j = 1, \dots, p$, et :

- $B_i^l = B_{i-1}^l$ et $R_i^l = R_{i-1}^l, \forall l \neq k$
- $B_i^k = \begin{cases} B_{i-1}^k \cup \{t_i\}, & \text{si } R_{i-1}^k \geq r_i^j \\ B_{i-1}^k, & \text{sinon} \end{cases}$
- $R_i^k = 0$ si $R_{i-1}^k < r_i^j$

En effet, le cas $R_{i-1}^k < r_i^j$ représente une situation de l'exécution dans laquelle la totalité des ressources disponibles R_{i-1}^k est consommée sans que la tâche t_i soit complètement achevée. Par conséquent, les ressources disponibles après cette exécution sont $R_i^k = 0$, et le gain $g_{a_k}(t_i)$ ne sera pas obtenu. Cela justifie la non appartenance de t_i à l'ensemble B_i^k dans cette situation. La probabilité de la transition entre l'état S_{i-1} et un état S_i est la probabilité que la quantité de ressources r_i^j soit consommée, donnée par $PE(t_i, r_i^j)$.

4.3 Le gain espéré et la politique optimale

Le choix d'une action à appliquer sur un état dépend du gain que le système espère obtenir en appliquant cette action. Nous notons par $Q(AL(t_i, a_k))$ le gain espéré obtenu de l'application de l'action $AL(t_i, a_k)$, et nous formalisons ce gain à la fin de ce paragraphe. Étant dans un état S_{i-1} , une politique $\pi(S_{i-1})$ à suivre est une des actions $AL(t_i, a_k)$, $a_k \in A$, à appliquer. Le gain espéré d'une politique $\pi(S_{i-1}) = AL(t_i, a_k)$ est celui correspondant à l'action appliquée : $Q(AL(t_i, a_k))$. Le gain espéré $V[S_{i-1}]$ associé à S_{i-1} est celui de la politique suivie dans cet état. Une politique optimale $\pi^*(S_{i-1})$ est toute politique qui maximise le gain espéré pour chaque état. Enfin, le calcul du gain espéré associé à chaque état peut être exprimé par les équations de Bellman que nous pouvons résoudre grâce à la valeur itérative [11], comme suit :

- pour un état non-terminal S_{i-1} , $i = 1, \dots, n$:

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

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

où, S_i^j est l'état correspondant à la consommation de la quantité r_i^j (Voir section 4.2).

- pour un état terminal $S_n = ((B_n^1, R_n^1), \dots, (B_n^m, R_n^m))$:

$$V[S_n] = \sum_{a_k \in A} \sum_{t_i \in B_n^k} g_{a_k}(t_i) \quad (3)$$

Vue la façon de calculer le gain espéré (équation 1) et que le PDM obtenu est à horizon fini sans boucles, alors la politique est optimale [1]. Formellement, la politique optimale $\pi^*(S_{i-1}); i = 1, \dots, n$ peut être définie comme suit :

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

L'équation précédente représente l'allocation optimale étant dans un état non-terminal S_{i-1} . La répartition optimale des tâches est obtenue à partir de l'état terminal du système, $S_n = ((B_n^1, T_n^1), \dots, (B_n^m, T_n^m))$, atteint par l'application de la politique optimale à partir de l'état initial S_0 . Dans notre contexte, l'exécution des tâches commence une fois la phase de l'allocation est à terme. Il est donc nécessaire de changer l'état du système après chaque application de la politique optimale. Nous expliquons dans ce qui suit comment effectuer ce changement d'état. Soit S_{i-1} l'état actuel du système et soit $\pi^*(S_{i-1}) = AL(t_i, a_k)$ la politique optimale calculée grâce à l'équation (4). L'application de $\pi^*(S_{i-1})$ sur l'état S_{i-1} conduit à un des p états possibles (section 4.2). L'état S_i qui sera l'état suivant de S_{i-1} est celui dont le gain espéré est maximal. D'une façon formelle :

$$S_i = \arg(\max \{PE(t_i, r_i^j) \times V[S_i^j]\})$$

où S_i^j est l'état correspondant à la consommation de la quantité r_i^j . L'état suivant de S_i sera déterminé de la même façon, et ainsi de suite.

Par ailleurs, le développement d'un PDMG est une opération coûteuse car le nombre d'états possibles est égale à $\frac{1-(m \times p)^{n+1}}{1-(m \times p)}$. Cette complexité est due au fait que la décision d'allouer une tâche à un agent est prise par un seul agent (le central) qui doit avoir toutes les informations sur le comportement incertain des autres (système centralisé). Afin de répartir les tâches dans un système décentralisé en tenant compte de l'exécution incertaine, nous proposons dans ce qui suit un mécanisme de répartition des tâches fondée sur la coordination des choix locaux des agents. En effet, chaque agent sélectionne les tâches qui maximisent son gain espéré via son PDML. Bien entendu, le développement d'un PDML par un agent est moins coûteux qu'un PDMG. Les agents communiquent ensuite afin de coordonner leur choix locaux. Dans cet objectif, deux stratégies de coordination seront introduites. Elles sont fondées sur l'allocation séquentielle des tâches pendant plusieurs cycles de négociation et distinctes par le nombre de tâches allouées et de messages échangés à chaque cycle de négociation.

5 LA SÉLECTION LOCALE DE TÂCHES COMME UN PDML

Dans cette section, nous formalisons la sélection locale des tâches en un processus décisionnel de Markov local. En effet, la sélection des tâches peut être vue comme un *processus décisionnel séquentiel*. À chaque étape i , l'agent décide s'il doit *exécuter* ou *ignorer* la tâche t_i . L'étape suivante concerne la décision (*exécuter* ou *ignorer*) pour la tâche suivante, et ainsi de suite. Le choix d'une décision (*exécuter* ou *ignorer*) ne dépend que de l'état actuel du processus décisionnel, c'est-à-dire : les tâches déjà choisies à exécuter et les ressources disponibles, ce qui fait de ce processus un processus Markovien.

Par la suite, nous détaillons la construction d'un PDML adapté à la sélection locale des tâches.

5.1 Le processus décisionnel de Markov local

Le PDML consiste en un ensemble d'états \bar{S} , un ensemble d'action \bar{AC} et en un modèle de transition. Nous associons à chaque état un gain espéré que le PDML maximise à long-terme. Nous décrivons dans la suite le PDML par : (1) les états, (2) les actions, (3) le modèle de transition et (4) le gain espéré localement.

5.1.1 La représentation des états

Chaque état de \bar{S} représente une situation de la *sélection* de tâches et de la *consommation anticipée de ressources* de l'agent (contrairement au PDMG où l'état représente la totalité des agents). Nous notons par $s_i = (B_i, R_i)$ l'état de l'agent à l'instant i , tel que :

- B_i est l'ensemble de tâches choisies à exécuter par l'agent jusqu'à l'instant i . La sélection d'une tâche se fait après l'anticipation de la quantité des ressources à consommer lors de l'exécution de cette tâche (Voir la section suivante). L'ensemble B_i vérifie :

$$B_i \subseteq \{t_1, \dots, t_i\}$$

- R_i sont les ressources disponibles.

La construction du PDML commence de l'état initial :

$$s_0 = (\emptyset, R_{a_k})$$

À l'instant n , l'agent arrive à un état terminal :

$$s_n = (B_n, T_n)$$

qui représente la situation finale de la sélection de tâches et de la consommation anticipée de ressources, c'est-à-dire : l'agent a effectué des décisions pour toutes les tâches ou toutes les ressources sont supposées être consommées. La transition d'un état à un autre se fait par l'application d'une action.

5.1.2 Les actions et le modèle de transition

Dans notre contexte, l'ensemble $\bar{\mathcal{A}}\mathcal{C}$ contient deux actions, à savoir :

- l'action $Ex(t_i)$ signifie que l'agent décide de sélectionner t_i à exécuter ;
- l'action $Ig(t_i)$ signifie que l'agent décide d'ignorer t_i .

L'action Ex comprend la sélection d'une tâche ainsi que l'anticipation de la quantité de ressources qui peut être consommée lors de l'exécution de cette tâche (action probabiliste). En effet, l'application de l'action $Ex(t_i)$ sur un état $s_{i-1} = (B_{i-1}, R_{i-1})$ conduit l'agent à un des états suivants :

$$s_i = (B_i, R_i = R_{i-1} - r_i^j)$$

où $j = 1, \dots, p$ et

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

$$- R_i = 0 \text{ si } R_{i-1} < r_i^j$$

En effet, le cas $R_{i-1} < r_i^j$ représente la même situation que nous avons vu dans la section (4.2). De même, la probabilité de la transition entre l'état s_{i-1} et un état s_i , en appliquant l'action $Ex(t_i)$ bien entendu, est de $PE(t_i, t_i^j)$.

L'action Ig quant à elle n'engendre aucune consommation de ressources car aucune tâche ne sera exécutée par l'agent (action déterministe). Étant dans un état $s_{i-1} = (B_{i-1}, R_{i-1})$, l'application de l'action $Ig(t_i)$ conduit l'agent à l'état :

$$s_i = (B_i = B_{i-1}, R_i = R_{i-1})$$

avec une probabilité égale à 1.

5.1.3 Le gain espéré et la politique optimale

La décision de choisir une action à appliquer (exécuter une tâche ou l'ignorer) dépend du gain que l'agent espère obtenir s'il applique cette action. Nous notons par $Q(Ex(t_i))$ et $Q(Ig(t_i))$ les deux gains espérés obtenus de l'application de l'action $Ex(t_i)$ et de l'action $Ig(t_i)$, respectivement. Étant dans un état s_{i-1} , une politique $\pi(s_{i-1})$ à suivre est une des deux actions $Ex(t_i)$ et $Ig(t_i)$ à appliquer. Le gain espéré d'une politique $\pi(s_{i-1})$ est celui de l'action appliquée. Le gain espéré $V[s_{i-1}]$ associé à $s_{i-1}, i = 1, \dots, n$ est celui de la politique suivie dans cet état. Une politique optimale $\pi^*(s_{i-1})$ est toute politique qui maximise le gain espéré pour chaque état. Enfin, le calcul du gain espéré associé à un état dans le PDML peut être exprimé par les équations de Bellman que nous pouvons résoudre grâce à la valeur itérative [11], comme suit :

- pour un état non-terminal $s_{i-1} = (B_{i-1}, R_{i-1}), i = 1, \dots, n$:

$$V[s_{i-1}] = \max\{Q(Ex(t_i)), Q(Ig(t_i))\} \quad (5)$$

$$Q(Ex(t_i)) = \sum_{j=1}^p PE(t_i, r_i^j) \times V[s_i^j] \quad (6)$$

$$Q(Ig(t_i)) = V[s_i = (B_{i-1}, R_{i-1})] \quad (7)$$

où, s_i^j dans l'équation (6) est l'état correspondant à la consommation de la quantité r_i^j , (Voir la section précédente) ;

- pour un état terminal $s_n = (B_n, R_n)$:

$$V[s_n] = \sum_{t_i \in B_n} g_{a_k}(t_i) \quad (8)$$

La politique optimale $\pi^*(s_{i-1})$ peut être définie comme dans la section (4.3) :

$$\pi^*(s_{i-1}) = \arg(\max\{Q(Ex(t_i)), Q(Ig(t_i))\}) \quad (9)$$

L'équation précédente représente l'action optimale à appliquer sur l'état non terminal s_{i-1} .

5.1.4 La sélection optimale des tâches

Nous appelons un choix optimal de l'agent l'ensemble de tâches $B_{\max} \subseteq T$ qui maximise son gain espéré. En d'autres termes, une tâche t_i appartient à B_{\max} si et seulement si la politique optimale exige son exécution.

Définition 4

On note par $\Pi(s_{i-1})$ la suite d'actions que l'agent doit appliquer, à partir de l'état s_{i-1} , afin de maximiser son gain espéré. Formellement,

$$\Pi(s_{i-1}) = \pi^*(s_{i-1}), \pi^*(s_i), \dots, \pi^*(s_{n-1})$$

où l'état $s_i, i = 1, \dots, n$, est l'état atteint par l'application de l'action $\pi^*(s_{i-1})$ sur l'état s_{i-1} .

Dans la définition précédente, la détermination de l'état s_i se réalise de la manière suivante :

- calculer $\pi^*(s_{i-1})$ selon l'équation (9);
- $s_i = \begin{cases} (B_i = B_{i-1}, R_i = R_{i-1}) & , \text{ si } \pi^*(s_{i-1}) = Ig(g_i); \\ \arg(\max_j \{PE(t_i, r_i^j) \times V[s_i^j]\}) & , \text{ sinon} \end{cases}$

où s_i^j est l'état correspondant à la consommation de la quantité r_i^j , (Voir section 5.1.2). Enfin, le choix optimal à partir d'un état s_{i-1} est l'ensemble $B_{\max}(s_{i-1})$ tel que :

$$B_{\max}(s_{i-1}) = \{t_j : Ex(t_j) \in \Pi(s_{i-1}), j = i, \dots, n\} \quad (10)$$

Le calcul d'un choix optimal par un agent est une opération moins coûteuse que la construction d'un PDMG. En effet un agent calcule son PDML d'une façon individuelle, c'est-à-dire sans avoir de connaissance des comportements incertains des autres agents. De plus, le nombre d'états possibles dans le PDML est considérablement moins élevé que dans le PDMG : $\frac{1-(p+1)^{n+1}}{1-(p+1)} < \frac{1-(m \times p)^{n+1}}{1-(m \times p)}$.

Cependant, le problème qui reste à résoudre consiste à coordonner les choix locaux afin d'éviter les conflits entre les agents (deux agents sélectionnent la même tâche). Comment coordonner les choix locaux fera l'objectif de la section suivante.

6 LA COORDINATION DES CHOIX LOCAUX

Dans notre système, la répartition des tâches est effectuée d'une façon séquentielle en plusieurs cycles de négociation entre les agents. En effet, à chaque cycle de négociation une ou plusieurs tâches sont allouées. Les tâches non allouées sont négociées dans le cycle suivant. Dans ce qui suit, nous présentons deux stratégies de coordination des choix optimaux. Elles réalisent l'allocation de tâches d'une façon séquentielle et elles tiennent compte du fait que les tâches allouées à un agents doivent appartenir à son choix optimal.

6.1 La stratégie Tâche_Par_Tâche (TPT)

Cette stratégie ressemble au mécanisme de la vente aux enchères. Les agents enchérissent, par les valeurs $g(t)$, sur une seule tâche pendant chaque cycle de négociation. Considérons un instant $i - 1$ où chaque agent est dans un état s_{i-1} dans son PDML. Un agent a_k envoie aux autres la valeur $g_{a_k}(t_i)$ si la politique optimale de son PDML exige l'exécution de t_i , c'est-à-dire $\pi^*(s_{i-1}) = Ex(t_i)$. Par convention, l'agent envoie la valeur 0 si $\pi^*(s_{i-1}) = Ig(t_i)$. Après la réception des valeurs $g_{a_k}(t_i)$, $a_k \in A$ par les agents, la tâche t_i est allouée à l'agent ayant proposé le gain maximal. Le cycle suivant de négociation commence par le changement d'état dans les PDMLs. En effet, l'agent auquel t_i a été allouée applique, dans son PDML, l'action $Ex(t_i)$. Comme cette action est probabiliste, l'agent anticipe ¹ son état futur s_i de la même façon que dans la section (5.1.4) :

$$s_i = arg(\max_j \{PE(t_i, r_i^j) \times V[s_i^j]\}) \quad (11)$$

Les autres agents appliquent, chacun dans son PDML, l'action $Ig(t_i)$ qui conduit à l'état $s_i = (B_i = B_{i-1}, R_i = R_{i-1})$. La tâche t_{i+1} sera ensuite négociée et allouée de la même manière que t_i , et ainsi de suite. La négociation se termine quand toutes les tâches sont allouées ou quand tous les agents ne peuvent plus choisir des tâches à exécuter (leurs ressources ne sont plus suffisantes).

¹cela est nécessaire dans les applications où l'exécution ne commence que après la fin de l'allocation

6.2 La stratégie Paquet_Par_Paquet (PPP)

Dans cette stratégie, un agent peut enchérir sur plusieurs tâches dans chaque cycle de négociation. Nous notons par Lib l'ensemble de tâches non allouées après chaque cycle de négociation. Dans le premier cycle, toutes les tâches sont non allouées $Lib = T$ et les agents sont, chacun dans son PDML, dans l'état s_0 . Chaque agent calcule son choix optimal $B_{max}(s_0)$, selon l'équation (10), et envoie ensuite les valeurs $g(t_l), \forall t_l \in B_{max}(s_0)$ aux autres agents. Après la réception de ces valeurs, les agents peuvent allouer chaque tâche t_l à l'agent ayant proposé le gain maximal. Le cycle suivant de négociation commence, s'il y a encore des tâches non-allouées, par le changement d'état dans les PDMLs et la mise-à-jour de l'ensemble Lib . Pour ce faire, chaque agent $a_k \in A$ change son état en vérifiant l'allocation des tâches l'une après l'autre, comme suit :

- $\forall t_l \in Lib$:
- si t_l est allouée à a_k , alors ce dernier élimine de son PDML le sous-espace d'états engendré à partir de l'action $Ig(t_l)$ et il applique ensuite l'action $Ex(t_l)$. L'état s_l produit par cette action peut être anticipé selon l'équation (11). $Lib = Lib - \{t_l\}$;
- si t_l est allouée à un autre agent, alors a_k élimine de son PDML le sous-espace d'états engendré à partir de l'action $Ex(t_l)$. Il applique ensuite l'action $Ig(t_l)$ qui le conduit à l'état $s_l = (B_l = B_{l-1}, R_l = R_{l-1})$. $Lib = Lib - \{t_l\}$.

Le nouvel état s_i , duquel le calcul d'un nouveau choix optimal commence, est celui qui vérifie la condition suivante :

$$(\forall t_{l \leq i}, t_l \notin Lib) \wedge (t_{i+1} \in Lib)$$

À partir de leur nouvel état s_i , les agents calculent à nouveau les ensembles $B_{max}(s_i)$. Les valeurs $g_{a_k}(t_l)$, où $t_l \in B_{max} \cap Lib, a_k \in A$, sont ensuite échangées entre les agents, et ainsi de suite. Il est possible qu'une (ou plusieurs) tâche $t_{l > i+1} \in B_{max} \cap Lib$ soit déjà allouée. Mais cela n'a aucun effet sur le choix optimal B_{max} de l'agent, ni sur l'allocation effectuée dans les cycles précédents. En effet, une tâche $t_{l > i+1}$ allouée à l'agent a_k apparaît dans son ensemble $B_{max}(s_i)$ car l'agent a déjà éliminé l'action $Ig(t_l)$. Pour un autre agent $a_{h \neq k}$, la tâche t_l ne peut pas appartenir à son ensemble $B_{max}(s_i)$ car l'action $Ex(t_l)$ a été exclue du PDML de a_h . Enfin, la négociation se termine quand toutes les tâches sont allouées ou quand tous les

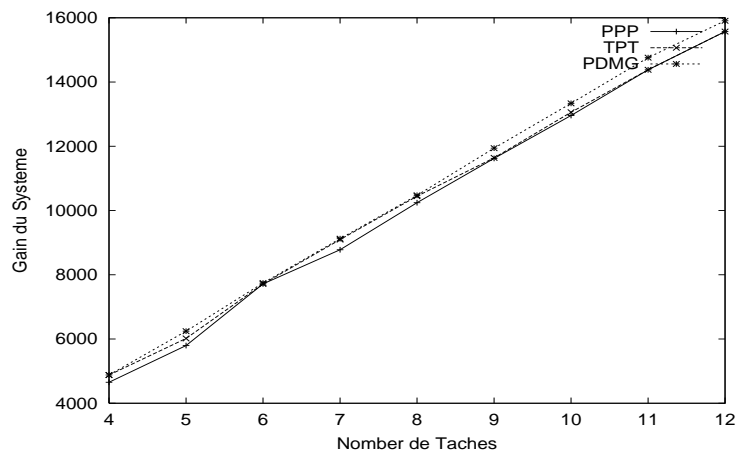


FIG. 1 – Le gain réel obtenu par le système en fonction de stratégie et comparé par le gain obtenu selon un PDMG.

agents ne peuvent plus choisir des tâches à exécuter (leurs ressources ne sont plus suffisantes).

7 RÉSULTATS EXPÉRIMENTAUX

L'allocation séquentielle suivie par les deux stratégies se réalise en un nombre fini de cycles de négociation. En effet, dans la stratégie TPT si les tâches non allouées ont été toutes mises en négociation l'une après l'autre et aucune d'elles n'a été enchérie, car les ressources restantes ne sont plus suffisantes, alors l'allocation se termine. En ce qui concerne la stratégie PPP, l'allocation se termine quand aucun agent n'enchérit sur aucune tâche non allouée. Afin d'étudier la performance des stratégies, nous les avons implémenté pour le problème décrit dans l'exemple présenté en section (2). Nous avons considéré un système de 3 robots planétaires et de différents nombres de tâches. La valeur de p est supposée égale à 3. Nous avons calculé le gain réel, $\sum_{a_k \in \mathcal{A}} \sum_{g \in B} r_{a_k}(g)$ où B est l'ensemble de tâches allouées à a_k , que le système peut obtenir en allouant les tâches selon chaque stratégie.

Tâches	Tâche_par_Tâche		Paquet_par_Paquet	
	Cycles	Messages	Cycles	Messages
4	4	24	1	6
5	5	30	1	6
6	6	36	2	12
7	7	42	2	12
8	8	48	2	12
9	9	54	2	12
10	10	60	2	12
11	11	66	3	18
12	12	72	3	18

FIG. 2 – Cycles de négociation et messages échangés en fonction du nombre de tâches et d'agents.

La figure (1) montre le gain réel obtenu selon les deux stratégies ainsi selon le PDMG. Nous remarquons que le gain obtenu selon la stratégie TPT est le plus proche de celui obtenu par PDMG (qui fournit l'allocation optimale pour un système centralisé). Cependant, la stratégie PPP permet une interaction entre les agents moins élevée que TPT. En effet, cette dernière engendre plus des messages entre les agents car n cycles de négociation sont demandés pour allouer n tâches, (voir le tableau sur la figure (2)).

8 DISCUSSION

Le problème de l'allocation des tâches dans un système multi-agent a été étudié en utilisant différentes techniques de coordination. Le protocole proposé dans [5] est basé sur le fait que les agents disposent d'une quantité illimitée de ressources. Cependant, dans notre contexte cette hypothèse n'est pas vérifiée. Des méthodes sont proposées dans [16, 17] pour réaliser l'allocation de tâches via la formation des coalitions. Elles traitent le cas où un agent est incapable d'exécuter une tâche d'une façon individuelle. Contrairement à notre approche, où chaque agent peut exécuter plusieurs tâches par lui-même. L'allocation centralisée a été déjà étudiée pour des applications comme la vente aux enchères combinatoire. Dans [12], l'allocation optimale des tâches exige que chaque agent enchérisse sur toutes les combinaisons de

tâches. Une grande quantité de données est donc envoyée par chaque agent au central. Les méthodes proposées dans [14, 15] diminuent le nombre de combinaisons à étudier par le central et accélèrent la coordination. Ces méthodes sont fondées sur le fait que le central tient compte uniquement des combinaisons sur lesquelles au moins un agent a enchéri. Dans notre contexte, l'agent enchérit sur les tâches une à une (pas de combinaisons), ce qui facilite la coordination. En effet, dans [12, 14] le central traite une quantité importante d'information (2^m combinaisons de tâches), alors que dans les deux stratégies proposées chaque agent n'en traite que m (le nombre d'agents) au pire des cas.

Par ailleurs, la notion du PDM a été utilisée afin de réaliser l'allocation des tâches dans un système non coopératif. Dans [3], l'allocation incertaine est considérée, alors que dans [8] un PDM est utilisé pour formaliser la sélection optimale des tâches où l'allocation et l'exécution sont incertaines. Dans un système centralisé coopératif, l'allocation optimale des tâches peut être réalisée via un PDMG [4]. Évidemment, la construction d'un PDMG dans un système décentralisé est une opération coûteuse malgré l'existence des méthodes qui permettent de réduire la complexité du calcul [6, 10, 9]. Ce qui justifie le besoin des mécanismes mieux adaptés pour les systèmes décentralisés.

9 CONCLUSION

Dans cet article, nous avons traité le problème de l'allocation des 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 but du système est d'allouer les tâches d'une façon maximisant son gain total en tenant compte de l'incertitude sur l'exécution de tâches. Afin de tenir compte de l'incertitude, chaque agent choisit le sous-ensemble de tâches qui maximise son gain espéré. En effet, la sélection des tâches maximisant le gain espéré est formalisée en un processus décisionnel de Markov. Ce qui permet à l'agent d'effectuer un choix optimal. Nous avons introduit deux stratégies de coordination des choix locaux des agents dans l'objectif de permettre au système de maximiser son gain. Les deux stratégies TPT et PPP, permettent aux agents d'allouer les tâches d'une façon séquentielle et de réviser leur choix locaux. La première stratégie TPT

engendre une interaction plus élevée que PPP, mais assure un gain réel supérieur. Cependant, la stratégie PPP propose un bon compromis entre les messages échangés et le gain obtenu.

Le travail à venir concernera l'utilisation de l'apprentissage par renforcement afin de diminuer la communication entre les agents. Cette technique permet à chaque agent d'apprendre, à partir des expériences précédentes, les tâches pour lesquelles il est considéré plus prioritaire pour l'exécution que les autres agents. Pour chaque tâche, l'agent peut améliorer ses connaissances sur la consommation des ressources.

RÉFÉRENCES

- [1] R. E. Bellman. A markov decision process. *journal of Mathematical Mechanics*, pages 6 :679–684, 1957.
- [2] S. Daniel Bernstein, Shlomo Zilberstein et 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.
- [3] Craig Boutilier, Moisés Goldszmidt et Bikash Sabata. Sequential auction for the allocation of resources with complementarities. In *Proceedings of IJCAI 99, Stockholm*, volume 1, pages 527–534, 1999.
- [4] Maroua Bouzid, Hosam Hanna et Abdel-Ilah Mouaddib. *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, chapitre 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] Davis et Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, pages 63–109, 1983.
- [6] Thomas Dean et Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1121–1127, 1995.

- [7] T. Estlin, A. Tobias, G. Rabideau, R. Castana et E. Mjolsness. An integrated system for multi-rever scientific exploration. In *Proceedings of AAAI 99*, pages 613–613, 1999.
- [8] Hosam Hanna et Abdel-Allah Mouaddib. Task selection problem under uncertainty as decision-making. In *Proceedings of International Joint conference on Autonomous Agents & Multi-Agent Systems, AAMAS02*, volume 3, pages 1303–1308, 2002.
- [9] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean et 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 et R.M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8) :1131–1147, 1998.
- [13] Thomas Sandholm. Contract types for satisficing task allocation : I theoretical results. In *Proceedings of AAAI 1998 Spring Symposium : Satisficing Models*, pages 68–75, 1998.
- [14] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of IJCAI 99, Stockholm*, pages 542–547, 1999.
- [15] Tuomas Sandholm. emediator : A next generation electronic commerce server. In *International Conference on Autonomous Agents (AGENTS), Barcelona, Spain*, june 2000.
- [16] Onn Shehory et Sarit Kraus. Task allocation via coalition formation among autonomous agents. In *Proceedings of IJCAI 95*, pages 655–661, Montreal, 1995.
- [17] Onn Shehory et Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101 :165–200, 1998.