Modèles pour la planification multi-agents de tâches complexes

Jules Dubanet[1], Josselin Guéneron[1]

[1] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

prenom.nom@unicaen.fr

Résumé

Nous proposons dans cet article deux nouveaux modèles pour la planification multi-agents de tâches simples et complexes. Le premier modèle s'appuie sur une exploration gloutonne de l'espace d'états, tandis que le deuxième modèle intègre un processus de négociation. Nous avons ensuite comparé nos deux modèles à un processus décisionnel markovien décentralisé partiellement observable résolu à l'aide de l'algorithme Multi-Agent Value Iteration. Nous avons principalement montré que nos modèles atteignent une optimalité comparable avec une résolution plus rapide.

Mots-clés

Systèmes multi-agents, Négociation, Allocation de tâches, Planification

Abstract

In this article, we propose two new models for multi-agent planning of simple and complex tasks. The first model is based on greedy exploration of the state space, while the second model incorporates a negotiation process. We then compared our two models to a decentralized partially observable Markov decision process solved using the Multi-Agent Value Iteration algorithm. We mainly showed that our models achieve comparable optimality with faster resolution.

Keywords

Multi-agent systems, negociation, task allocation, planning

1 Introduction

Les agents autonomes, à l'instar des drones sont utilisés de manière croissante dans de nombreux domaines comme le militaire, la logistique, les réseaux électriques intelligents ou encore la protection civile [3]. La gestion des interactions entre ceux-ci présente un véritable défi que l'étude des systèmes multi-agents tente de relever, en particulier lorsque les agents doivent se coordonner et coopérer [8]. À titre d'exemple, une des applications critiques des systèmes multi-agents coopératifs est le *disaster response* (gestion de catastrophe). Dans un problème de *disaster response*, un ensemble d'agents autonomes, parfois hétérogènes dans leurs compétences et dans leur nature (UAV, UAG, parfois humains), doit coopérer et se coordonner afin de répondre à une catastrophe, telle que des inondations ou un séisme. Ce problème se déroule donc dans un environnement dangereux et incertain en raison du caractère imprévisible des catastrophes et de leurs conséquences. Par conséquent, les tâches à réaliser nécessitent souvent plusieurs compétences, les rendant complexes.

Plusieurs approches existantes permettent de formaliser ces systèmes multi-agents. Parmi elles nous retrouvons les processus de décision markoviens (MDP) et leurs extensions multi-agents [1]. Ces processus présentent de nombreux avantages, ils sont facilement modélisables et gèrent très bien la stochasticité de l'environnement. Cependant, ils sont limités dans la représentation de tâches complexes nécessitant la coopération de plusieurs agents.

Nous pouvons également évoquer le formalisme des formations de coalitions qui propose de regrouper les agents en sous-groupes nommés "coalitions", leur permettant ainsi de mettre en commun leurs compétences afin de coopérer pour effectuer des tâches complexes [5]. Ce formalisme est donc tout à fait adapté aux environnements complexes. Il existe déjà des méthodes de planification basées sur la formation de coalitions, mais ces travaux ne prennent en compte aucune stochasticité, les tâches étant supposées parfaitement réalisables et les gains obtenus par les agents certains. Or, si l'on s'intéresse au contexte de la *disaster response*, l'environnement est rempli d'incertitudes et il est donc nécessaire de pouvoir proposer des modèles capables de traiter des tâches complexes dont la réalisation est stochastique, et ce, de manière répétée. De plus, la formation de coalitions est souvent coûteuse à résoudre en termes de temps et mémoire.

En prenant en compte ces restrictions, le domaine de l'*allocation de tâches multi-agents* (MRTA, *multi-robot task allocation*) semble prometteur. En effet, la gestion de catastrophes implique de devoir assigner dynamiquement des tâches critiques à des agents aux compétences variées, en tenant compte à la fois des incertitudes liées à l'environnement et des contraintes de coopération, ce qui correspond très bien à ce qui est réalisable grâce à la MRTA. Les approches existantes présentent donc des limites : les modèles de MDP ne prennent pas en compte des tâches complexes, tandis que les modèles de formation de coalitions peinent à intégrer de l'incertitude [6] et sont coûteux. L'enjeu est donc de développer un modèle où les agents peuvent planifier la réalisation de tâches complexes, en possible coopération, dans un contexte distribué ou décentralisé. Dans cet article, nous nous intéressons à cette notamment à dernière piste en nous basant sur la MRTA, et nous présenterons des

pistes d'ouvertures vers l'intégration du dynamisme et de la stochasticité dans nos modèles.

Cet article est structuré comme suit. Dans la section 2, nous présenterons les différents formalismes de la littérature sur lesquels nous nous appuyerons, avant de présenter nos contributions en section 3. Des expérimentations seront présentées en section 4 avant de conclure et donner des perspectives en section 5.

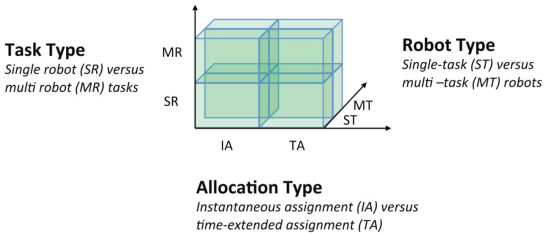


FIGURE 1 – Taxonomie de l'allocation de tâches [7]

2 État de l'art

Dans cette section nous allons présenter les éléments de littérature utiles à la compréhension des modèles développés, à savoir l'allocation de tâches multi-robots, les processus de décision markovien, et leur extension décentralisée et partiellement observable qui nous serviront d'élément de comparaison.

2.1 Allocation de tâches

Dans le cadre des systèmes multi-agents, l' *allocation de tâches* est une catégorie de problèmes dans lesquels des agents doivent réaliser des tâches au sein d'un environnement. Le but de ces problèmes est de faire émerger entre les agents de la coordination voire de la coopération. Cette coordination peut être implicite (aussi appelée émergente) quand elle résulte d'une somme d'interactions locales entre les agents et l'environnement, mais elle peut aussi être explicite (ou intentionnelle) quand les agents sont explicitement assignés à certaines tâches. Ces problèmes peuvent être catégorisés selon 3 axes [4] : le type de tâche, le type d'agent, le type d'allocation. Pour chacun de ces axes, il y a 2 choix possibles. Respectivement il y a :

- **Single-robot Task (ST)** ou **Multi-robot Task (MT)**, Single-robot Task signifie que les tâches n'ont besoin que d'un seul agent pour être complétées alors que les Multi-robot Task nécessitent plusieurs agents ;
- **Single-task Robot (SR)** ou **Multi-task Robot (MR)**, Single-task Robot signifie que les agents ne peuvent exécuter qu'une seule tâche à la fois tandis que Multi-task Robot signifie que certains agents peuvent faire plusieurs tâches à la fois ;
- **Instantaneous assignement (IA)** ou **Time-extended assignement (TA)**, Instantaneous assignement signifie que le modèle ne permet que l'allocation instantanée des agents sans prévisions sur de futures allocations, tandis que dans le cas Time-extended allocation, de la prévision est possible, par exemple en fournissant l'ordre d'arrivée de nouvelles tâches.

La figure 1 résume graphiquement ces axes. Grâce à cette taxonomie, nous pouvons catégoriser des classes de problèmes, par exemple si nous qualifions un problème de ST-MR-IA, cela signifie que les tâches ne nécessitent qu'un agent, que les agents peuvent faire plusieurs tâches à la fois et que l'allocation se fait selon les informations dont nous disposons sur l'instant.
Un problème d'allocation de tâches multi-robots (MRTA) est donc un problème de coordination où un groupe d'agents doivent efficacement s'assigner à des tâches afin de les exécuter.

Définition 1 (Problème de MRTA) *Formellement, un problème de MRTA est défini par un tuple* $\langle R, T, u \rangle$ *tel que :*

- $R = r_1, r_2, ..., r_n$*, un ensemble de* n *robots,*
- $T = t_1, t_2, ..., t_m$*, un ensemble de* m *tâches,*
- $u : R \times T \to \mathbb{R}$*, une fonction d'utilité qui associe à chaque couple (robot, tâche) une valeur correspondant au gain du robot si celui-ci effectue la tâche.*

Nous notons donc u_{ij} *l'utilité gagnée (ou le coût infligé dans le cas d'une utilité négative) par le robot* r_i *réalisant la tâche* t_j*.*

Parmi les très nombreuses manières de résoudre les problèmes de *task allocation*, les algorithmes basés sur le principe d'enchères (*Auction based algorithms*) sont particulièrement intéressants [2]. Le principe général est que les agents peuvent miser sur les tâches à accomplir et s'ils remportent la mise, ils sont alloués à la tâche. Ces mises peuvent dépendre de plusieurs paramètres, un des plus courant étant la capacité de l'agent à effectuer la tâche, et plus il est compétent pour une tâche, plus il pourra enchérir dessus. Un autre paramètre commun est la distance à la tâche (pouvant être associée à un coût de déplacement). L'algorithme CBAA (*Consensus-Based Auction Algorithm*) se base sur ce principe. Il se déroule en 2 phases. Premièrement, les agents communiquent une enchère pour la tâche qui est la plus avantageuse de leur point de vue. Les agents mettent ensuite à jour une liste associant les tâches avec l'agent qui a formulé la meilleure enchère. Cette première phase est la phase d'enchères, la deuxième phase qui la suit est la phase de consensus. Les agents partagent leur liste avec leurs voisins et pour chaque tâche, ils gardent l'offre la plus compétitive. Ce processus est répété itérativement et converge vers un état où tous les agents ont la même liste.

2.2 Processus de Décision Markovien

Un processus de décision markovien (MDP) est un modèle stochastique de la théorie de la décision permettant de modéliser la prise de décision d'un agent dans un environnement incertain [9].

Définition 2 (Processus de Décision Markovien)
Un processus de décision markovien est un tuple $\mathcal{M} = \langle S, A, T, R \rangle$ *avec :*

- S *un ensemble d'états,*
- A *un ensemble d'actions réalisables par l'agent,*
- $T(s'|s, a)$ *une fonction de transition qui renvoie un état en fonction de l'état courant et de l'action effectuée par l'agent,*
- R *une fonction de récompense qui prend en entrée l'état courant et une action réalisée par l'agent* $R(s, a)$.

La résolution d'un MDP consiste à déterminer une *politique* optimale (*i.e.* maximisant l'espérance de gains).

Définition 3 (Politique) *Une* politique *décrit l'action à effectuer par l'agent dans chaque état possible du MDP. Mathématiquement, il s'agit d'une fonction* $\pi : S \rightarrow A$ *qui à chaque état associe une action. Une politique peut être déterministe (*$\pi(s) = a$*) ou stochastique (*$\pi(a|s) = P(a|s)$*). La politique optimale* $\pi*$ *est calculée à partir de l'équation de Bellman :*

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} [R(s, a) + \gamma V^*(s')] T(s, a, s')$$

avec $\gamma \in [0, 1]$ *un facteur d'atténuation.*

Les DEC-POMDP (*Decentralized Partially Observable MDP*) sont une extension des MDP permettant de considérer plusieurs agents évoluant dans un même environnement partiellement observable [1]. Les agents possèdent des croyances individuelles sur l'environnement, qu'ils affinent à partir de nouvelles observations, et calculent une politique individuelle à partir de leurs croyances.

Définition 4 (DEC-POMDP) *Un DEC-POMDP est un tuple*
$\langle S, \{A_1, .., An\}, T, R, \{\Omega_1, .., \Omega_n\}, \{O_1, ..O_n\} \rangle$ *avec :*

- S *un ensemble d'états,*
- $\{A_1, .., A_n\}$ *l'ensemble des actions de chaque agent,*
- $T(s'|s, a)$ *une fonction de transition,*
- R *une fonction de récompense,*
- $\{\Omega_1, .., \Omega_n\}$ *un espace d'observation pour chaque agent,*
- $\{O_1, .., O_n\}$ *une fonction d'observation propre à chaque agent.*

Ces agents agissent de manière décentralisée afin de calculer une politique jointe, c'est-à-dire une politique qui prend en compte les actions de l'ensemble des agents.

3 Contributions

Dans cet article, nous proposons deux modèles. Ces deux modèles partagent certaines caractéristiques : les agents agissent de manière coopérative, l'exécution des actions des agents se fait de manière cyclique, et les agents ont le moins d'informations possible. En effet dans le contexte du *disaster response*, les communications sont parfois limitées (les infrastructures de communication peuvent être endommagées, ou les services saturés). La coopération est essentielle car, d'une part tous nos agents poursuivent le même but, à savoir la complétion des tâches, et d'autre part, la plupart des situations réelles peuvent nécessiter diverses compétences qu'un seul agent ne peut pas toujours posséder. Le premier modèle prend en considération des tâches dites simples (c'est-à-dire qui ne nécessitent qu'une seule compétence pour être réalisées), et utilise une méthode d'exploration gloutonne. Pour ces raisons, nous appellerons ce premier modèle le modèle simple ou à exploration. Le deuxième modèle quant à lui prend en compte des tâches plus complexes (c'est-à-dire qui nécessitent une combinaison de compétences pour être réalisées) et utilise un processus de négociation. Nous l'appellerons donc le modèle complexe ou le modèle à négociation. En utilisant la taxonomie des modèles de *task allocation*, le modèle simple peut se noter ST-SR-IA et le modèle complexe MT-SR-IA. Bien que les contraintes de précédence entre tâches, le dynamisme et la stochasticité sont de fort intérêt dans le contexte du *disaster response*, les intégrer directement aux modèles aurait été complexe, c'est pourquoi nous nous sommes concentrés dans cet article sur des versions statiques, déterministes et sans contraintes de précédence.

3.1 Éléments communs aux deux modèles

Les deux modèles présentés dans cette section utilisent des mécanismes de décision différents mais sont fondés sur des caractéristiques et concepts de modélisation similaires. Dans les deux modèles, nous avons un ensemble d'agents qui ont pour objectif de réaliser des tâches afin de gagner des récompenses. Pour ce faire, ils peuvent s'associer aux tâches : l'association de tous les agents forme un état.

Définition 5 (Environnement des modèles)
L'environnement de nos modèles est défini par un tuple $\langle N, J, C, r \rangle$ *tel que :*

- $N = \{n_1, n_2, ..., n_l\}$ *: l'ensemble des agents,*
- $J = \{j_1, j_2, ..., j_m\}$ *: l'ensemble des tâches,*
- $C = \{c_1, c_2, ..., c_k\}$ *: l'ensemble des compétences, avec* $\forall i \in [1, k],\ c_i \in \mathbb{R}$
- $u : J \rightarrow \mathbb{R}$ *: la fonction d'utilité associant à chaque tâche une utilité réelle.*

Les agents du modèles peuvent s'associer à une tâche, nous pouvons donc définir :

- j_n *: la tâche associée à un agent* n,
- $N_j = \{n \mid n \in N\ t.q. j_n = j\}$ *: l'ensemble des agents associés à la tâche* j.

Définition 6 (Tâches) *Les tâches représentées dans les modèles doivent être réalisées par les agents. Elles pos-*

sèdent des besoins spécifiques :

$$B_j = \{c_1^j, c_2^j, ..., c_k^j\}$$

À leur réalisation, les tâches fournissent donc une utilité aux agents responsables de leur réalisation, cette utilité est notée u_j.

La fonction de récompense (différente de la fonction d'utilité) est spécifique à chaque modèle et sera donc présentée dans les parties dédiées aux modèles.

Définition 7 (Agent) *Chaque agent possède un certain nombre de compétences, notées :*

$$C_n = \{c_1^n, c_2^n, ..., c_k^n\}$$

Avec $C_n \subseteq C$. *Ces compétences permettent de réaliser les tâches en fonction de leurs besoins.*

Définition 8 (Action d'un agent) *L'action d'un agent* n *est définie comme le fait de se positionner sur une tâche. Les actions sont définies de la même manière dans les deux modèles. L'ensemble des tâches réalisables par l'agent* a *se définit comme :*

$$A_n = \{j \mid j \in J \; \exists c_i \in C, \; t.q. \; B_j(c_i) \leqslant C_n(c_i)\}$$

S'il existe au moins un besoin d'une compétence dans les besoins de j *où l'agent* n *est suffisamment compétent, il pourra être alloué à* j.

Un agent possède donc autant d'actions possibles que de tâches réalisables en adéquation avec son vecteur de compétences. Pour chaque agent, la taille de l'espace d'action maximal est $|J|$.

Définition 9 (État des modèles) *Dans les modèles, un état est défini par les ensembles d'agents associés à chaque tâche. Nous avons donc :*

$$s = \{j_1 : \{n_1, n_2...\}, .., j_m : \{n_{l-1}, n_l\}\}$$

L'espace d'états est constitué de l'ensemble des allocations entre tâches et agents. Nous pouvons également considérer un état comme étant une action jointe de tous les agents, pour cette raison l'espace d'états est le produit de l'espace d'actions de tous les agents.

Exemple 1

- *Soit* $N = \{n_1\}$ *un ensemble d'agents de taille 1,*
- *Soit* $A_{n_1} = \{j_1, j_2, ..., j_m\}$ *l'ensemble des tâches réalisables par* n_1,

Ici, il n'y a qu'un seul agent, l'espace d'états se résume donc à l'espace des tâches auxquelles il peut s'associer. Nous avons donc $|S| = |A_n| = m$.

Nous pouvons en déduire que pour un système avec l agents dont les actions sont bornées par J, l'espace d'états est $|J|^l$.

Définition 10 (Social Welfare) *Le social welfare, ou bien-être social, est une mesure globale de la satisfaction des agents, c'est une fonction qui prend en entrée les récompenses individuelles des agents et retourne un indicateur de la qualité du système. Ici, la fonction d'agrégation utilisée est la somme. Nous avons donc :*

$$SW(s) = \sum_{n \in N} R(s, n)$$

Cette fonction est commune aux deux modèles où seule la définition de $R(s, n)$ change. C'est également cette fonction qui est utilisée dans la formalisation sous forme de DEC-POMDP pour calculer la récompense de chaque état.

3.2 Modèle à exploration

Le premier modèle présenté est un modèle de *task allocation* de type ST-SR-TA, c'est-à-dire que les agents ne peuvent s'associer qu'à une seule tâche et que les tâches ne nécessitent qu'un agent pour être réalisées. Les agents utilisent une exploration et une évaluation gloutonne des états, par conséquent, ils vont explorer les états qu'ils sont en mesure d'atteindre selon leurs connaissances de l'environnement. Ils connaissent l'état actuel, c'est-à-dire l'allocation tâches-agents courante, ainsi que leurs propres compétences desquelles ils peuvent déduire les tâches sur lesquelles ils peuvent se positionner. L'exécution du modèle s'articule ainsi :

1. Les agents calculent une politique pour chaque état qu'ils considèrent atteignable et pour l'état actuel.
2. Les agents appliquent leur politique puis en calcule une nouvelle pour les nouveaux états atteignables.

Les états que les agents prennent en compte sont uniquement ceux où les autres agents ne bougent pas, puisque les agents ne considèrent que leurs propres compétences. De ce fait, les agents se retrouvent dans des états qu'ils n'avaient pas anticipés, étoffant ainsi leurs connaissances des états possibles. Le calcul d'une politique se fait en regardant tous les mouvements possibles à partir d'un état (c'est-à-dire envisager de se positionner sur toutes les tâches réalisables) et de garder l'action qui mène à l'état maximisant la récompense.

3.2.1 Fonction de récompense

Les récompenses sont calculées grâce à une fonction et permettent d'influencer le comportement des agents. Elle est définie comme suit :

$$R(s, n) = \frac{u_j c_j^n}{N_j} \prod_{j' \in A_n} [N_{j'} + \epsilon]$$

Avec :

- s un état,
- n un agent,
- j la tâche à laquelle l'agent n est associé,
- u_j l'utilité associée à la tâche j,
- c_j^n la compétence de n pour le besoin de j,
- N_j le nombre d'agents associés à j (ce nombre vaut toujours au moins 1 car quand un agent calcule la

récompense d'un état, il se projette dans celui-ci et calcule la récompense liée à l'association avec j, donc si personne n'était déjà associé à j, N_j vaudra 1),
— $\epsilon \in]0, 1[$ un terme évitant la présence de 0 dans le produit.

Le produit des $N_{j'} + \epsilon$ sert à favoriser les cas de répartition égale entre les tâches et à pénaliser les cas où une ou plusieurs tâches seraient laissées sans agents associés. Puisque ϵ est plus petit que 1, si un $N_{j'}$ est à 0, le produit final sera impacté et nullifié. La valeur maximale de ce produit est atteinte quand la répartition entre les tâches est parfaitement égale. Nous allons démontrer cette affirmation ci-dessous.

Démonstration 1 *Prenons un exemple : soient deux tâches, et $2x$ agents que nous répartissons sur les deux tâches. Nous souhaitons démontrer qu'en cas de répartition égale des agents, le produit du nombre d'agents associés à chaque tâche est supérieur au produit du nombre d'agents associés à chaque tâche en cas de répartition inégale. Nous avons donc deux situations :*

1. *Répartition inégale : respectivement $x - y$ et $x + y$ pour la première et deuxième tâche.*
2. *Répartition égale : x pour chaque tâche.*

Nous pouvons donc formuler une inéquation dont nous devons vérifier la cohérence, avec à gauche le terme représentant une répartition inégale, et à droite, une répartition égale :

$$(x - y)(x + y) \leq x \times x$$

Nous avons bien la somme des termes de chaque côté de l'équation qui est égale à $2x$ (notre nombre d'agents). Tentons de développer l'inéquation. Selon les identités remarquables, nous avons l'égalité suivante.

$$(a + b)(a - b) = a^2 - b^2$$

Nous pouvons donc développer le terme de gauche de notre inéquation.

$$x^2 - y^2 \leq x^2$$

L'inéquation est donc correcte. Elle est également généralisable à davantage de termes de façon similaire. Nous pouvons donc affirmer que lorsque le nombre de termes est égal, ainsi que leur somme, le produit des termes lorsque la répartition est égale est supérieur à celui obtenu avec une répartition inégale.

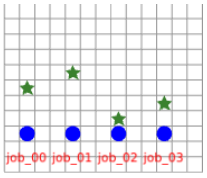
Ce produit se fait entre toutes les tâches visibles par a (noté A_a). Cette restriction permet de diminuer les calculs et ne change rien quant aux résultats.

Exemple 2 *Par exemple, prenons :*
— *n_1 un agent,*
— *j_1, j_2 et j_3 des tâches,*
— *$A_n = \{j_1, j_2\}$.*

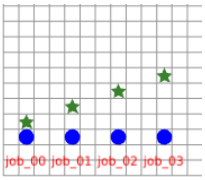
L'agent fera en sorte que la répartition entre j_1 et j_2 soit égale, mais l'état de j_3 n'influencera pas sa décision. Si nous prenons en compte toutes les tâches, l'état de j_3 changerait la valeur de la récompense, mais pas la dynamique entre les états (à savoir est-ce qu'un état a une récompense plus élevée qu'un autre).

Avec cette fonction de récompense, les agents vont chercher à maximiser leur récompense individuelle, mais elle est maximale dans un état où les agents sont bien répartis, les agents vont donc prioriser ces états au lieu de tous se positionner sur la tâche ayant la récompense la plus élevée.

La récompense dépend en partie de l'expertise de l'agent pour la tâche, ainsi plus un agent est compétent pour une tâche, plus il va préférer celle-ci, comme il peut-être observé sur la figure 2.



(a) Compétences homogènes



(b) Expertise des agents

FIGURE 2 – Exemple de répartition après exécution du modèle simple avec 4 agents et 4 tâches. Dans ces images, les agents (étoiles vertes) évoluent dans une grille et se placent au dessus des tâches (disques bleus) avec lesquelles ils s'associent. Leur positionnement en vertical est égal à leur indice + 1.

La figure 2a (gauche) présente une répartition des agents où leurs compétences sont homogènes et la figure 2b (droite) où ils sont experts pour la compétence qui a le même nom ou numéro qu'eux. Les agents sont placés de manière à ce que leur coordonnée (en ordonnée) est égale à l'ordonnée de la tâche additionnée à leur numéro (+1 car la numérotation commence à partir de 0, cela empêche donc les chevauchements). Nous constatons donc que les agents se positionnent sur les tâches dont ils sont experts. Il est important de noter que dans le cas où les compétences sont homogènes, toute configuration où il n'y a qu'un seul agent par tâche est optimale.

3.3 Modèle à négociation

Ce deuxième modèle permet de pallier une des limites du premier modèle : la non prise en compte de tâches plus complexes. Ces tâches sont dites complexes, car leurs besoins sont multiples et peuvent donc nécessiter une certaine diversité de compétences pour être complétées.

Définition 11 (Tâche complexe) *Une tâche est complexe lorsque son vecteur de besoin est de taille supérieur à 1 :*

$$|B_j| > 1$$

Une collaboration des agents est donc nécessaire dans ce modèle. L'approche gloutonne utilisée dans le premier modèle n'est pas adaptée à de la collaboration, car les agents doivent se mettre d'accord sur la tâche à réaliser là où avec l'approche gloutonne les agents se déplacent et agissent de manière égoïste.

C'est pourquoi nous utiliserons un processus de négociation entre les agents. Cette négociation se fait grâce à un négociateur, par exemple un agent tiré aléatoirement. Le seul prérequis du négociateur est d'être en communication avec l'ensemble des autres agents, à la fois en réception et en emission. Cette communication peut-être directe ou indirecte. Le but de la négociation est de trouver un état qui satisfait tous les agents, par conséquent les agents doivent pouvoir exprimer des préférences sur les états. Cette méthode de négociation n'engendre pas de conflit car l'état à atteindre est calculé par le négociateur qui cherche à atteindre un consensus, mais en cas d'égalité, la préférence du négociateur prévaudra sur les autres.

Définition 12 (Vecteur de préférence) *Pour représenter les préférences sur les états des agents, les agents possèdent un vecteur de préférence défini tel que :*

$$V_n = \{s \in S \mid s \succeq s' ssi\ R(s,n) \geq R(s',n)\}$$

Ce vecteur de préférence est limité en taille par un paramètre d, lors de la construction de ce vecteur chaque agent parcourt tous les états possibles et les ajoute au vecteur si c'est possible (c'est-à-dire si le vecteur est de taille inférieure à d ou si le dernier état du vecteur à une récompense inférieure ou égale).

3.3.1 Fonction de récompense

La fonction de récompense utilisée pour construire ce vecteur est différente de celle utilisée dans le premier modèle, mais s'appuie sur la même base. Elle est définie comme suit :

$$R(s,n) = \sum_{i=1}^{k} \left(\frac{u_j c_i^n}{N_j} \left(\frac{|b_{sat}^j|}{|B_j|} \right)^f \frac{1}{\sum_{q \in j} |C_n \cap C_q| \epsilon} \right)$$

Cette fonction est découpée en 3 grands termes :

- $\frac{u_j c_i^n}{N_j}$: très similaire à celui dans la fonction du premier modèle, où la seule différence est que c_j^n qui était la compétence requise unique de la tâche devient c_i^n
- $\left(\frac{|b_{sat}^j|}{|B_j|}\right)^f$: il s'agit du nombre de besoins satisfaits par les agents alloués à j $|b_{sat}^j|$ divisé par l'ensemble des besoins $|B_j|$. Ce terme rend donc les états où les tâches sont satisfaites plus attirants pour les agents.
- $\frac{1}{\sum_{l \in j} |C_n \cap C_q| \epsilon}$: ce terme est l'inverse de la somme de la taille des intersections des compétences entre l'agent n et les agents q qui sont également associés à la tâche j. Ce troisième terme évite la redondance des tâches au sein des agents présents sur j.

La grande différence avec la fonction de récompense du modèle simple est que le comportement des agents que l'on veut inciter *via* la fonction n'est pas le même. Dans le premier modèle pour les tâches simples, si un agent est associé à une tâche, c'est que celle-ci va pouvoir être réalisée, donc pour maximiser l'efficacité des agents, il faut les inciter à se repartir sur l'ensemble des tâches. Or ici, la présence d'un agent sur une tâche ne garantit pas sa faisabilité ; si nous voulons maximiser le nombre de tâches faites, il faut récompenser les agents quand l'ensemble des besoins de la tâche sont satisfaits. C'est exactement ce que fait le deuxième terme de la fonction $\left(\frac{|b_{sat}^j|}{|B_j|}\right)^f$. Le deuxième terme est modulé par f car dans certains cas, les deux premiers termes se compensent.

Exemple 3 *Par exemple, si un agent est seul et satisfait la moitié des besoins de j, nous avons :*

- $N_j = 1$
- $\frac{|b_{sat}^j|}{|B_j|} = 0.5$,

mais si un deuxième agent vient compléter les besoins, nous aurons

- $N_j = 2$
- $\frac{|b_{sat}^j|}{|B_j|} = 1$

Or, avec u_j fixe, nous avons :

$$\frac{u_j}{1} \times \frac{1}{2} = \frac{u_j}{2} \times \frac{2}{2}$$

Donc les récompenses des deux cas sont équivalentes alors que le deuxième cas est préférable car j est pleinement satisfaite.

Afin de pouvoir réaliser une tâche, les agents doivent être complémentaires dans leurs compétences et nous voulons donc encourager cette complémentarité et cela est fait grâce au troisième terme : $\frac{1}{\sum_{l \in j} |C_n \cap C_q| \epsilon}$. Le ϵ évite d'avoir une division par 0 lorsqu'un agent est seul ou parfaitement complémentaire. Grâce à ce terme, un comportement intéressant émerge : imaginons qu'il y ait 2 agents qui partagent une part de leurs compétences mais qu'ils n'ont pas, même a 2, les compétences requises pour satisfaire pleinement une tâche, alors ils vont se mettre sur 2 tâches différentes, préférant faire chacun une part d'une tâche plutôt que de se partager une part plus grande mais incomplète d'une seule tâche.
Tous les termes de cette fonction sont encapsulés dans une somme qui concerne les compétences de l'agent n, c'est-à-dire que la récompense est la somme pour chaque besoin de j qu'il est en mesure de satisfaire.
Comme le modèle simple, cette fonction de récompense dépend en partie des compétences de l'agent, et il préférera des tâches qui correspondent plus à son vecteur de compétences. Cependant, l'effet produit par l'introduction d'agents experts dans certaines compétences est moins impactant sur le résultat qu'avec le modèle simple, notamment à cause du fait que certaines tâches peuvent partager une compétence requise et que les tâches ne se résument pas à une seule compétence. Dans le cadre des tâches simples, quand un agent est expert pour une compétence, il sera expert pour toutes les tâches nécessitant cette compétence, les mettant donc largement en priorité face aux autres, alors que dans le cadre des tâches complexes pour qu'un agent soit expert sur une tâche il faut qu'il soit expert sur l'ensemble des compétences qui constituent ses besoins.

3.3.2 Processus de négociation

Le processus de négociation se déroule de la manière suivante : dans un premier temps, un négociateur initial est choisi aléatoirement par les agents. Ce négociateur initial est le point de départ du cycle de négociation (dans l'ordre des indices des agents par exemple), c'est-à-dire qu'à chaque nouvel appel du processus de négociation, c'est l'agent suivant dans le cycle qui sera le nouveau négociateur. Pour une itération de négociation, le négociateur prend son propre vecteur de préférence comme référence, et pour chaque état dans ce vecteur de référence, il calcule l'indice carré moyen de cet état parmi les vecteurs de préférence de chaque agent. Si l'état n'apparaît pas dans un vecteur de préférence, l'indice considéré sera $d + 1$. Une fois que tous les états du vecteur de référence ont été parcourus, l'état dont l'indice carré moyen est le plus petit est désigné comme l'état négocié qui sera transmis à tous les agents. Chaque agent s'associera à la tâche en adéquation avec cet état. En cas d'égalité entre plusieurs états, c'est l'état préféré du négociateur parmi ceux-ci qui sera choisi. L'algorithme 1 donne sa version en pseudo-code, pour un appel avec le négociateur courant.

Algorithme 1 : Algorithme de Négociation

Données :
Ensemble des agents N, ensemble des états S, vecteurs de préférences V_n pour chaque agent $n \in N$.
Cycle de négociation $\mathcal{C} = \{0, \ldots, |N| - 1\}$, indice du négociateur courant l

Début

$n \leftarrow \mathcal{C}_l$
$V_{ref} \leftarrow V_n$
pour *chaque état* $s \in V_{ref}$ **faire**
  $sqr\_sum(s) \leftarrow 0$
  pour *chaque agent* $n \in \mathcal{N}$ **faire**
    si $s \in V_n$ **alors**
      $i \leftarrow$ indice de s dans V_n
    sinon
      $i \leftarrow d + 1$
    $sqr\_sum(s) \leftarrow sqr\_sum(s) + i^2$
  $MSI(s) \leftarrow \frac{1}{|N|} sqr\_sum(s)$
$s^* \leftarrow \arg\min_{s \in V_{ref}} MSI(s)$
$l \leftarrow (l + 1) \bmod |N|$

Transmettre s^* à tous les agents $n \in N$
Chaque agent n s'associe à la tâche allouée par s^*

L'algorithme de négociation a une complexité en $O(Nd^2)$. Le paramètre d est donc très important, car il contrôle la difficulté computationnelle, mais également le degré d'optimalité de l'algorithme. En effet, si d est trop petit, il y a moins de chance que le négociateur trouve un état qui fasse consensus et il prendra un état du vecteur de référence. À l'inverse, si d est grand, la négociation sera longue. Il faut donc trouver un équilibre entre optimalité et rapidité, nous suivons donc l'intuition telle que d doit être strictement supérieur au nombre d'états optimaux du système, mais des travaux sont encore nécessaires pour mieux comprendre la gestion et l'impact de ce paramètre.

4 Expérimentations

Nous avons précédemment présenté nos modèles, leurs spécificités et leur fonctionnement. Nous allons maintenant nous intéresser à leurs performances. Afin d'évaluer les performances de nos modèles, nous allons les comparer aux DEC-POMDP résolus par l'algorithme *Multi-Agent Value Iteration* (MA-VI), une des approches à l'état de l'art en planification, assurant des résultats optimaux. La politique jointe calculée par MA-VI est ensuite utilisée à la place des états alloués/négociés de nos modèles.

4.1 Résultats

Dans nos expérimentations, les seuls paramètres variants sont le nombres d'agents et le nombre de tâches.
Pour le modèle utilisant la négociation, le paramètre d a été fixé à 10 suite à des tests empiriques évaluant son influence, cette valeur ressortant comme étant pertinente. Les valeurs présentées dans le tableau 1 représentent le temps d'exécution moyen (sur 5 exécutions), en millisecondes, pour la complétion de toutes les tâches, pour chacun de nos modèles ainsi que MA-VI. La première colonne indique la configuration utilisée respectivement en termes d'agents et tâches. L'abréviation DNF signifie *did not finish* et indique que l'exécution n'a pas abouti avant un timeout préconfiguré d'une heure.

Config.	Simple	Comp.	MA-VI
2-2	53.5	110.7	138.9
2-4	80.7	234.6	735.2
4-4	81.5	3902.6	68708.1
6-6	136.2	DNF	DNF

TABLE 1 – Temps de calculs moyens

Nous constatons que le modèle simple, c'est-à-dire celui qui utilise une exploration gloutonne des états, est beaucoup plus rapide que le modèle complexe et également plus rapide que MA-VI. De plus, le modèle simple est capable de gérer un nombre de tâches et d'agents plus important, en témoigne le test sur la configuration 6 agents et 6 tâches où seul le modèle simple est parvenu à un résultat avant la fin du timeout. La rapidité du modèle simple peut s'expliquer facilement car à chaque cycle, chaque agent explore au plus $|J|$ états, donc la complexité ne dépend pas du nombre d'agents mais du nombre de tâches. La scalabilité de ce modèle sur un système distribué est donc très grande (si le système n'est pas distribué, la complexité dépend quand même du nombre d'agents). Le modèle complexe, qui utilise la négociation, est quant à lui plus lent que le modèle simple (mais plus rapide que MA-VI) et ne parvient pas à résoudre des problèmes où le nombre d'états est très grand. En effet, le modèle complexe nécessite un calcul préalable de tous les états, par conséquent, les calculs des vecteurs de

préférences se complexifient avec le nombre d'états. MA-VI, qui est plus lent que le modèle complexe, doit calculer une politique jointe sur l'ensemble des états et des agents, ce qui est extrêmement coûteux en termes de calculs, d'où sa lenteur.
Dans un second temps nous allons comparer l'optimalité des trois approches en utilisant 2 métriques : le social welfare et le nombre d'étapes pour réaliser toutes les tâches.

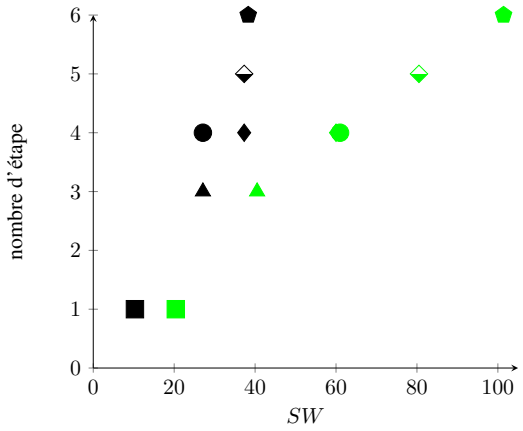


FIGURE 3 – Comparaison entre le modèle simple et MA-VI pour le social welfare et le nombre d'étapes avant complétion

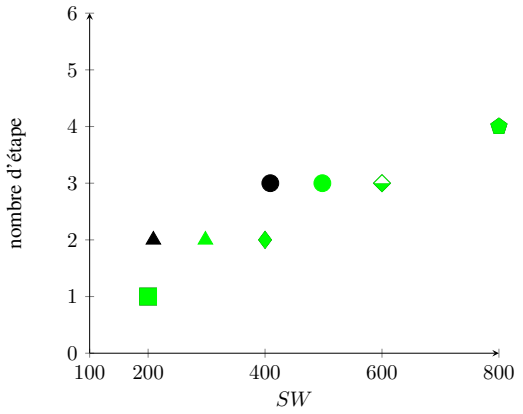


FIGURE 4 – Comparaison entre le modèle complexe et MA-VI pour le social welfare et le nombre d'étapes avant complétion

La figure 3 présente les résultats d'exprérimentations comparant le modèle simple avec MA-VI et la figure 4 présente les résultats d'exprérimentations comparant le modèle complexe avec MA-VI. Les 2 modèles ont été lancés sur des configurations contenant 2 agents et un nombre de tâches variant entre 2 (carré), 3 (triangle), 4 (losange), 5 (cercle), 6 (losange à moitié plein) et 8 (pentagone). Les résultats des modèles sont en noir tandis que ceux de MA-VI sont en vert. Ces résultats présentent le total accumulé de récompense (Social Welfare) en abscisse et le nombre d'étapes pour satisfaire toutes les tâches en ordonnée. Pour le modèle simple, nous constatons que MA-VI est plus optimal au sens du Social Welfare mais est équivalent pour le nombre d'étapes. De plus, ce gain en optimalité s'aggrandit avec le nombre de tâches disponibles. Pour le modèle complexe, les performances en termes de Social Welfare entre le modèle et MA-VI sont très similaires et seules les configurations avec un nombre impair de tâches (3 et 5) présentent une différence. Cela peut s'expliquer par le fait que lorsqu'il ne reste qu'une seule tâche, MA-VI va préferer mettre un agent seul sur la tâche. À l'inverse, les agents vont aller à 2 sur la tâche avec le modèle à négociation, ce qui se traduit par un Social Welfare inférieur.
En résumé, MA-VI dépasse le modèle simple en termes de Social Welfare, mais pour un temps de calcul beaucoup plus grand. Quant au modèle complexe, il est très similaire à MA-VI en termes de Social Welfare, toutefois, le modèle complexe est plus rapide, et la différence de rapidité s'accentue lorsque l'espace d'état grandit. Concernant le nombre d'étapes pour réaliser l'ensemble des tâches, tous les modèles sont équivalents.

5 Conclusion

Nous avons proposé dans cet article deux modèles pour la planification multi-agents de tâches. Ces modèles, fondés sur l'allocation de tâches, s'inscrivent dans un contexte où les agents ont des tâches simples ou complexes à accomplir au sein d'un environnement. Le premier modèle utilise une exploration gloutonne des états. Ce modèle est prometteur, toutefois, dû à son caractère glouton, nous supposons qu'il n'existe pas de garantie théorique de trouver un état optimal, la recherche pouvant s'arrêter sur un optimum local. Expérimentalement, l'optimalité en termes de nombre d'étapes a été atteinte à chaque fois, cependant, en termes de Social Welfare, ce modèle est légèrement en dessous. Il offre cependant une grande scalabilité ainsi qu'une grande rapidité d'exécution. Ce modèle est dédié à la résolution de problèmes ne contenant que des tâches simples, cela limite grandement les possibilités de modélisation, les tâches nécessitant l'intervention de plusieurs agents ne sont pas modélisables.
Le second modèle est quant à lui pensé pour les tâches complexes, nécessitant l'intervention de plusieurs agents. Il utilise un mécanisme de négociation et permet ainsi la coopération entre les agents. Le mécanisme de négociation de ce modèle nécessite de connaître l'ensemble des états possibles, et calculer l'ensemble des états est très vite coûteux, en témoigne les temps de calculs qui sont plus importants que ceux du modèle simple, mais reste toutefois plus rapide que MA-VI, pour une optimalité en termes de nombre d'étapes et de Social Welfare similaire. Une limite de ce modèle est liée au paramètre d, qui contrôle la taille des vecteurs de préférences des agents, et qui agit donc sur la complexité du modèle. Plus spécifiquement, la complexité de l'algorithme de négociation dépend de d, et donc si ce dernier doit être grand pour garantir l'optimalité, l'algorithme de négociation sera lent.
Les pistes de recherches futures concernent principalement les extensions à des contextes dynamiques et stochastiques,

qui nécessiteront d'adapter les modèles à l'apparition et disparition des tâches, ainsi qu'à l'estimation des récompenses désormais stochastiques. Cela nécessitera en particulier d'adapter le processus de négociation afin de mettre à jour les vecteurs de préférence, mais également de s'appuyer sur des modèles d'apprentissage (en particulier par renforcement) afin de palier aux aspects stochastiques. L'integration de tâches temporellement contraintes (durée, expiration) est également une piste de réflexion afin de se rapprocher d'une situation applicative réelle. De plus, il serait intéressant de repenser les modèles afin que le mécanisme de décision ne s'appuie pas uniquement sur la fonction de récompense.

Références

[1] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4) :819–840, 2002.

[2] Han-Lim Choi, Luc Brunet, and Jonathan P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE transactions on robotics*, 25(4) :912–926, 2009.

[3] Ali Dorri, Salil S. Kanhere, and Raja Jurdak. Multi-agent systems : A survey. *IEEE Access*, 6 :28573–28593, 2018.

[4] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International journal of robotics research*, 23(9) :939–954, 2004.

[5] Josselin Guéneron and Grégory Bonnet. De la diversité des jeux de coalitions à utilité transférable. In *29es Journées Francophones sur les Systèmes Multi-Agents*, Bordeaux, France, June 2021.

[6] Josselin Guéneron and Grégory Bonnet. Un concept de solutions avec un biais d'exploration pour les jeux de coalitions stochastiques répétés. In *31es Journées Francophones sur les Systèmes Multi-Agents*, Strasbourg, France, July 2023.

[7] Ayorkor G. Korsah, Anthony Stentz, and M Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, 32 :12, 2013.

[8] Carla Mouradian, Jagruti Sahoo, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. A coalition formation algorithm for multi-robot task allocation in large-scale natural disasters. In *13th international wireless communications and mobile computing conference*, pages 1909–1914. IEEE, 2017.

[9] Laurent Péret and Frédérick Garcia. On-line search for solving markov decision processes via heuristic sampling. *Learning*, 16 :2, 2004.