Allocation décentralisée de lots par consensus pour des tâches composites multi-modes

Gauthier Picard
gauthier.picard@onera.fr

ONERA/DTIS, Université de Toulouse, France

Résumé

Nous considérons des agents planifiant des tâches à exécuter dans des créneaux temporels qu'ils possèdent sur des ressources disjonctives. Des utilisateurs non propriétaires souhaitent programmer des tâches composites multi-modes nécessitant l'accès à un ou plusieurs créneaux privés, dont les propriétaires doivent se coordonner pour répondre collectivement aux requêtes, sans divulguer leurs propres plans. Nous adoptons une approche d'allocation de tâches basée consensus, MM-CBGA, adaptée à ce cadre. La contribution est évaluée et comparée à des allocations centralisées gloutonnes et à des enchères séquentielles classiques, en utilisant des constellations d'observation de la Terre simulées et des carnets de commande réalistes pour des observations sur l'Europe.

Mots-clés : *Allocation de tâches, consensus, observation de la Terre*

Abstract

We consider agents scheduling tasks to be executed in time slots they own on disjunctive resources. Non-owner users wish to schedule multi-mode composite tasks requiring access to one or more private slots, whose owners must coordinate to collectively respond to requests, without disclosing their own plans. We adopt a consensus-based task allocation approach, MM-CBGA, adapted to this framework. The contribution is evaluated and compared to greedy centralized allocations and classical sequential auctions, using simulated Earth observation constellations and realistic order books for observations over Europe.

Keywords: *task allocation, consensus, Earth observation*

1 Introduction

Le déploiement de constellations de plus grandes tailles pour les tâches d'observation de la Terre nécessite la résolution de problèmes de planification et d'ordonnancement de plus en plus complexes. En effet, si un plus grand nombre de satellites augmente linéairement le nombre d'opportunités de satisfaire les demandes des utilisateurs à une fréquence plus élevée, cela augmente aussi exponentiellement la taille des problèmes d'ordonnancement et d'allocation de tâches qui en résultent [21]. De plus, la résolution de tels problèmes doit être effectuée plusieurs fois par jour, en moins de quelques minutes, afin d'adapter les plans aux conditions météorologiques, ou aux demandes de dernière minute, et afin que les plans soient envoyés aux satellites dès que possible. Pour rendre la situation encore plus complexe, certaines constellations envisagent désormais des concepts d'exploitation où certains utilisateurs possèdent certains créneaux orbitaux pour une longue période [19, 15, 16]. Les propriétaires gèrent entièrement les plans dans leurs créneaux orbitaux et les gardent privés. Tout utilisateur non propriétaire souhaitant effectuer certaines observations nécessitant l'accès à des créneaux orbitaux privés doit alors faire une demande aux propriétaires, qui doivent se coordonner puis décider d'accepter ou non ces tâches. Ce problème revient à trouver une allocation de tâches aux créneaux privés, afin que les demandes soient satisfaites de la meilleure manière, tout en minimisant leur impact sur les plans privés et sans les divulguer.

Ce problème entre dans le cadre de l'allocation multi-agents de tâches [7, 17]. Parmi les méthodes efficaces, les techniques basées enchères ont fait l'objet d'une attention particulière, et ont été utilisées dans les domaines de la robotique collective [11] et de l'Espace [13, 18]. Ici, les agents mettent en œuvre des enchères pour répartir les tâches de manière coopérative et distribuée. Les agents font des offres pour des tâches uniques ou des lots de tâches qu'ils veulent insérer dans leurs plans, et les gagnants sont déterminés en fonction de la valeur des offres, de manière centralisée (e.g., enchères combinatoires, PSI, SSI [12]) ou décentralisée (e.g., CBBA [2]). Dans le domaine de l'allocation de tâches d'observation, Picard a fourni des modèles et des algorithmes pour des scénarios spécifiques à l'observation de la Terre avec des créneaux privés [19], où chaque mode ne contenait qu'une seule tâche. Dans une direction légèrement différente, Phillips et Parra étudient l'utilisation de l'allocation de lots basée sur le consensus entre les satellites eux-mêmes, mais sans aucune confidentialité [18]. Enfin, Lee et al. considèrent plusieurs constellations appartenant à différents agents et proposent d'utiliser des protocoles de consensus

pour allouer les tâches au sein de ces constellations [13]. Dans tous ces modèles et algorithmes, les demandes composites ne sont pas prises en compte. Les tâches ne sont pas interdépendantes ou ne font pas partie d'une tâche composite : les agents ne se soucient que de la cohérence et de la valeur de leur lot, et non de certaines contraintes sur l'ensemble des tâches atomiques de la même tâche composite. Pour répondre à de telles interdépendances d'éléments, on peut s'appuyer sur une détermination centralisée du gagnant (ou *Winner Determination Problem*, WDP), garantissant que les contraintes ou les critères sur les tâches composites sont optimisés [14]. Dans des contextes plus distribués, tels que l'allocation de tâches par consensus (e.g., CBAA ou CBBA), où le WDP est distribuée, la capture des interdépendances entre les tâches n'est pas souvent prise en compte. Cependant, Hunt et al. ont proposé une extension de CBBA, appelée CBGA, pour traiter les tâches multi-agents, c'est-à-dire les tâches nécessitant la participation de plusieurs agents ; mais là encore, un seul mode a été considéré pour chacune de ces tâches.

Nous proposons ici une extension de CBGA pour faire face aux tâches composites multi-modes, et l'appliquons à un scénario d'observation de la Terre avec de multiples propriétaires de créneaux orbitaux. Nos contributions sont les suivantes : (i) nous modélisons le problème d'allocation de tâches composites multi-mode multi-agent (MACTA) ; (ii) nous modélisons MACTA comme un programme mathématique, et proposons un algorithme glouton pour le résoudre ; (iii) nous concevons un solveur distribué pour MACTA (MM-CBGA) ; (iv) nous faisons correspondre une version étendue du problème d'ordonnancement de constellations de satellites d'observation de la Terre [19] à MACTA, en considérant des tâches composites au lieu de tâches monoscopiques ; (v) nous évaluons les performances de MM-CBGA sur des carnets de commande réalistes et un simulateur de constellation.

Cet article est une traduction d'un article accepté à AAMAS'23 [20].

2 Modèle et notations

Cette section présente le problème d'allocation de tâches composites multi-agents abordé ici.

Définition 1. Un *créneau* $w = [s_w, e_w]$ est une fenêtre temporelle, où $s_w \in [t_{\min}, t_{\max}]$, $e_w \in [t_{\min}, t_{\max}]$, $s_w < e_w$, et $t_{\min} < t_{\max} \in \mathbb{R}_{\geq 0}$.

Soit $\mathcal{T}$ un ensemble de tâches à exécuter et $\mathcal{R}$ un ensemble de ressources *disjonctives*, i.e. qui ne peuvent être utilisées que pour une seule tâche à la fois. Dans notre article, les tâches seront des tâches d'observation à effectuer sur des satellites, qui ne peuvent effectuer qu'une seule observation à la fois.

Définition 2. Une *tâche* $\tau \in \mathcal{T}$ est un tuple $\langle w_\tau, d_\tau, r_\tau, \omega_\tau \rangle$ où $w_\tau = [s_\tau, e_\tau]$ est le créneau dans lequel τ doit être planifiée, $d_\tau \in \mathbb{R}_{\geq 0}$ est la durée de τ, $r_\tau \in \mathcal{R}$ est une ressource sur laquelle τ doit être exécutée, et $\omega_\tau \in \mathbb{R}_{\geq 0}$ est la récompense reçue pour avoir planifié τ.

Définition 3. Un *plan* est un ensemble $\pi = \{(\tau, t) \mid \tau \in \mathcal{T}, t \in [t_{\min}, t_{\max}]\}$ définissant le temps de début t de chaque tâche τ, tel qu'il n'y a aucun chevauchement entre tâches sur la même ressource : $\forall (\tau, t), (\tau', t'), [t, t+d_\tau] \cap [t', t'+d_{\tau'}] = \emptyset$.

Soit $\mathring{\mathcal{T}}$ l'ensemble des requêtes à réaliser. Dans notre étude, certaines requêtes sont émises par des clients non propriétaires et nécessitent l'accès à des créneaux privés.

Définition 4. Une *requête* (ou *tâche composite*) $\mathring{\tau} \in \mathring{\mathcal{T}}$ est un tuple $\langle M_{\mathring{\tau}}, \oplus_{\mathring{\tau}} \rangle$ où $M_{\mathring{\tau}} \subset 2^{\mathcal{T}}$ est l'ensemble d'ensembles de tâches pouvant compléter la requête (i.e. l'ensemble de *modes*), et $\oplus_{\mathring{\tau}} : M_{\mathring{\tau}} \to \mathbb{R}_{\geq 0}$ est une fonction d'agrégation qui calcule la récompense associée à chaque mode.

La notion de *mode* est courante dans la gestion de projet à contraintes de ressources (RCPSP) [3], et est même utilisée dans le domaine de l'observation de la Terre [21]. Il est pratique de modéliser les interdépendances entre les tâches atomiques pour répondre aux requêtes. La fonction d'agrégation $\oplus_{\mathring{\tau}}$ peut être n'importe quelle fonction monotone croissante telle que la somme (+), ou une fonction linéaire avec un certain facteur d'actualisation pour toute tâche après la première [1]. Une requête est satisfaite si l'un de ses modes est entièrement planifié, i.e. toutes les tâches d'un mode sont planifiées.

Définition 5. Pour un mode m d'une requête $\mathring{\tau}$, la *récompense de mode* est l'agrégation des récompenses des tâches composant le mode : $\omega_m \stackrel{\text{def}}{=} \oplus_{\mathring{\tau}}(m)$.

Pour une tâche donnée τ, on note $\mathsf{request}(\tau)$ (resp. $\mathsf{mode}(\tau)$) la requête (resp. le mode) à laquelle τ contribue. On note également $\mathsf{modes}(\mathring{\tau})$ l'ensemble de tous les modes pour la requête $\mathring{\tau}$, et par extension nous notons $\mathsf{modes}(T) = \bigcup_{\mathring{\tau}_i \in T} \mathsf{modes}(\mathring{\tau}_i)$.

Définition 6. Pour un plan π, la *récompense de plan*, notée ω_π, est la somme des récompenses des modes planifiés :

$$\omega_\pi \stackrel{\text{def}}{=} \sum_{m \in \{\mathsf{mode}(\tau) \mid \tau \in \pi\}} \omega_m$$

1. Pour des raisons de lisibilité et de simplicité, nous utiliserons la somme simple (+ et $\sum$), au lieu de l'opérateur spécifique à la requête ($\oplus^{\mathring{\tau}}$ et $\bigoplus^{\mathring{\tau}}$) dans le reste de l'article.

Soit $\mathcal{A}$ un ensemble d'agents capables d'effectuer des tâches. Dans nos expériences, les agents seront des propriétaires de créneaux privés, qui peuvent recevoir des demandes d'insertion de certaines observations dans leurs créneaux privés.

Définition 7. Un *agent* $i \in \mathcal{A}$ est un tuple $\langle O_i, \mathring{\mathcal{T}}_i^-, \mathring{\mathcal{T}}_i^+, \pi_i \rangle$ où $O_i = \{(r, w) \mid r \in \mathcal{R}, w = [s_w, e_w]\}$ est un ensemble de créneaux privés sur des ressources, $\mathring{\mathcal{T}}_i^- \subset \mathring{\mathcal{T}}$ est un ensemble de requêtes privées à positionner dans ses créneaux privés, $\mathring{\mathcal{T}}_i^+ \subset \mathring{\mathcal{T}}$ est un ensemble de requêtes externes pour lesquelles i est sollicité, et π_i est le plan courant de i qui affecte un temps de début à certaines tâches privées ou externes.

Notez qu'il n'y a pas de copropriété des créneaux, c-à-d. que pour la même ressource r, il n'y a pas de chevauchement entre les créneaux privés. De plus, nous supposons que les fenêtres des tâches privées sont entièrement incluses dans les créneaux privés. Nous notons également que π_i est divisé en deux sous-parties : π_i^- pour les tâches privées, et $\pi_i^+ = \pi_i \setminus \pi_i^-$ pour les tâches externes. Nous identifions les tâches qu'un agent peut exécuter, c-à-d. dont les fenêtres temporelles sont incluses dans certains de ses créneaux privés, en utilisant $\mathsf{can} : \mathcal{A} \to 2^{\mathcal{T}}$. Par extension, nous utilisons également can pour les tâches composites, si au moins une tâche peut être programmée dans l'un des créneaux privés de l'agent.

Définition 8. Un *problème d'allocation de tâches composites multi-modes multi-agents* (ou MACTA) est un tuple $P = \langle \mathcal{A}, \mathcal{R}, \mathring{\mathcal{T}} \rangle$, défini par un ensemble d'agents $\mathcal{A}$, un ensemble de ressources disjonctives $\mathcal{R}$, et un ensemble de requêtes $\mathring{\mathcal{T}}$ (privées ou non), et vise à trouver l'allocation de tâches atomiques aux agents, qui maximise la somme des récompenses des requêtes satisfaites tout en respectant les contraintes de disjonction entre les ressources : $\max \sum_{\mathring{\tau} \in \mathring{\mathcal{T}}} \omega_{\mathring{\tau}}$.

Pour un MACTA P donné, on note $\mathring{\mathcal{T}}^+ \stackrel{\text{def}}{=} \bigcup_{i \in \mathcal{A}} \{\mathring{\mathcal{T}}_i^+\}$ l'ensemble des tâches qui sont privées à aucun propriétaire. Pour un agent donné i, nous notons P_i^- le problème restreint à $\mathring{\mathcal{T}}_i^-$, afin de calculer son plan initial π_i ou de le réviser, compte tenu de certaines tâches supplémentaires. Nous notons également $P_i^- \uplus M$ le problème consistant à résoudre le problème privé P_i^- avec des modes supplémentaires contenus dans l'ensemble M, où chaque mode est filtré de façon à ne considérer que les tâches situées dans les créneaux de i. Il sera utilisé pour évaluer le coût de l'insertion des modes dans les plans.

Notez que, bien que les agents puissent avoir des plans privés initiaux, nous considérons que ces plans peuvent être révisés afin d'augmenter la récompense collective. Cependant, si certains agents sont réticents à réviser leur plan, ils peuvent soit toujours répondre "non" à toute demande chevauchant leurs créneaux, soit augmenter la récompense de leurs propres tâches et demandes pour garantir leur présence dans le plan final.

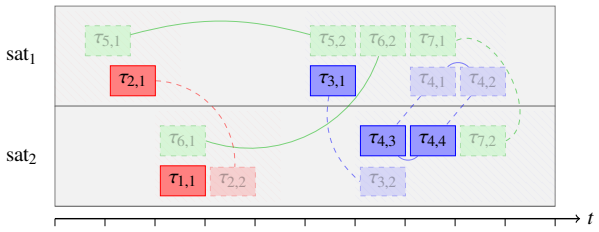

FIGURE 1 – Un exemple de MACTA. Les créneaux privés sont hachurés en rouge (u_1) ou en bleu (u_2). Les tâches sont représentées par des rectangles colorés pleins lorsque planifiées, en pointillés lorsque non planifiées. Les tâches d'une même requête sont liées par des lignes pleines pour le même mode, ou par des pointillées pour des modes distincts.

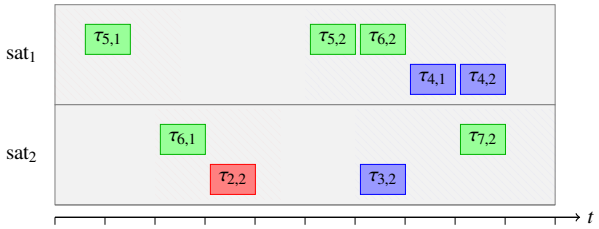

FIGURE 2 – Une solution pour l'exemple 1.

Exemple 1. La figure 1 illustre un MACTA simple. Nous considérons l'horizon $[t_{\min}, t_{\max}] = [0, 10]$, et une constellation de deux satellites, $\mathcal{R} = \{\text{sat}_1, \text{sat}_2\}$. Deux agents, $\mathcal{A} = \{u_1, u_2\}$, possèdent deux créneaux chacun, émettent deux requêtes privées, et ont déjà calculé leurs plans initiaux. Ainsi, u_1 a émis $\mathring{\tau}_1$ avec un unique mode $\{\tau_{1,1}\}$, et $\mathring{\tau}_2$ avec deux modes $\{\tau_{2,1}\}$ et $\{\tau_{2,2}\}$. u_2 a émis deux requêtes, $\mathring{\tau}_3$ avec deux modes $\{\tau_{3,1}\}$ et $\{\tau_{3,2}\}$, et $\mathring{\tau}_4$ avec deux modes $\{\tau_{4,1}, \tau_{4,2}\}$ et $\{\tau_{4,3}, \tau_{4,4}\}$. En particulier, la requête "tout-ou-rien" $\mathring{\tau}_4$ nécessite de planifier deux tâches, ou aucune.

Considérons les récompenses suivantes $\omega_{\tau_{1,1}} = \omega_{\tau_{2,1}} = \omega_{\tau_{4,3}} = \omega_{\tau_{4,4}} = 10$, $\omega_{\tau_{2,2}} = \omega_{\tau_{3,1}} = \omega_{\tau_{3,2}} = \omega_{\tau_{4,1}} = 8$, $\omega_{\tau_{4,2}} = 6$. En utilisant la fonction d'agrégation +, les meilleurs plans privés sont représentés dans la figure 1, sélectionnant les meilleurs modes pour chaque requête, et avec une récompense globale de 48.

Maintenant, lorsque des requêtes externes sont émises, les plans privés peuvent changer radicalement afin d'augmenter la récompense globale. Considérons trois nouvelles requêtes (en vert sur les figures) émises par un utilisateur externe, avec

toutes la même récompense de tâche atomique de 6 : $\mathring{\tau}_5$ avec un mode $\{\tau_{5,1},\tau_{5,2}\}$, $\mathring{\tau}_6$ avec un mode $\{\tau_{6,1},\tau_{6,2}\}$, et $\mathring{\tau}_7$ avec deux modes $\{\tau_{7,1}\}$ et $\{\tau_{7,2}\}$. Sans réviser les plans initiaux de u_1 et u_2, la seule demande externe satisfaite est $\mathring{\tau}_7$, en plaçant $\tau_{7,2}$, ce qui entraîne une récompense globale de $48+6=54$.

u_1 gagne à supprimer la tâche $\tau_{1,1}$ au profit de $\tau_{6,1}$ et $\tau_{6,2}$, pour un gain de $6+6-10=2$, mais nécessite de se coordonner avec u_2 qui doit insérer $\tau_{6,2}$ dans son plan. u_1 remplace $\tau_{2,1}$ par $\tau_{2,2}$, laissant de la place pour $\tau_{5,1}$, et nécessitant de planifier $\tau_{5,2}$, ce qui oblige u_2 à remplacer $\tau_{3,1}$ par $\tau_{3,2}$, ce qui génère finalement un gain de $6+6-10+8-8=2$. La récompense du plan solution illustré dans la figure 2, qui est optimale, est 60.

3 Approches centralisées

Une approche classique pour résoudre de manière centralisée un problème d'allocation consiste à utiliser un programme mathématique – idéalement linéaire – afin de bénéficier de solveurs efficaces disponibles, tels qu'IBM CPLEX [10] ou Gurobi [8]. Il est également possible de s'appuyer sur des algorithmes gloutons, afin de passer à l'échelle. Notez que ces solutions ne respectent évidemment pas la privacité : les agents doivent envoyer toutes leurs données à une autorité centrale chargée de calculer les horaires. Cependant, elles représentent de bonnes bases de comparaison pour évaluer la qualité des solutions obtenues de manière décentralisée, et elles peuvent être utilisées par les agents pour calculer ou évaluer des plans privés.

Approche optimale. Nous formulons MACTA comme un programme linéaire mixte en nombres entiers (MILP). Considérons les variables de décision suivantes : $x_\tau \in \{0,1\}$ une variable binaire indiquant si la tâche τ est planifiée ; $y_m \in \{0,1\}$ une variable binaire indiquant si le mode m est sélectionné pour répondre à une requête $\mathring{\tau}$; $t_\tau \in [s_{w_\tau}, e_{w_\tau} - d_\tau]$ une variable continue indiquant la date de début de τ ; et $\beta_{\tau,\tau'} \in \{0,1\}$ une variable binaire indiquant si τ précède τ'.Nous notons $(\tau,\tau') \in \mathcal{T}_\cap^2$ les paires de tâches distinctes sur la même ressource avec des fenêtres de temps qui se chevauchent.

$$\max_{y_m} \sum_{\mathring{\tau}\in\mathring{\mathcal{T}}} \sum_{m\in M_{\mathring{\tau}}} \omega_m y_m \tag{1}$$

$$\text{t.q.} \sum_{\tau\in m} x_\tau \geq |m| y_m, \quad \forall \mathring{\tau} \in \mathring{\mathcal{T}}, \forall m \in M_{\mathring{\tau}} \tag{2}$$

$$\sum_{m\in M_{\mathring{\tau}}} y_m \leq 1, \quad \forall \mathring{\tau} \in \mathring{\mathcal{T}} \tag{3}$$

$$2-\beta_{\tau,\tau'}-\beta_{\tau',\tau} \leq x_\tau, \quad \forall (\tau,\tau') \in \mathcal{T}_\cap^2 \tag{4}$$

$$2-\beta_{\tau,\tau'}-\beta_{\tau',\tau} \leq x_{\tau'}, \quad \forall (\tau,\tau') \in \mathcal{T}_\cap^2 \tag{5}$$

$$\beta_{\tau,\tau'}+\beta_{\tau',\tau} \leq 3-x_\tau-x_{\tau'}, \quad \forall (\tau,\tau') \in \mathcal{T}_\cap^2 \tag{6}$$

$$t_\tau - t_{\tau'} \geq d_\tau - \Delta^{\max}_{\tau,\tau'}\beta_{\tau,\tau'}, \quad \forall (\tau,\tau') \in \mathcal{T}_\cap^2 \tag{7}$$

$$t_{\tau'} - t_\tau \geq d'_\tau - \Delta^{\max}_{\tau',\tau}\beta_{\tau',\tau}, \quad \forall (\tau,\tau') \in \mathcal{T}_\cap^2 \tag{8}$$

avec $\Delta^{\max}_{\tau,\tau'} = e_\tau - s_{\tau'} + d_\tau$ une valeur servant de constante *big-M* pour déclencher des contraintes conditionnées par la précédence entre deux tâches. Ce MILP vise à maximiser la récompense des modes actifs, comme indiqué dans l'équation (1). La contrainte (2) relie les x et les y en indiquant que si un mode est sélectionné, toutes ses tâches doivent être programmées. La contrainte (3) oblige à sélectionner au maximum un mode par requête. Les contraintes (4) à (6) assurent la cohérence des variables de précédence, tandis que les contraintes (7) et (8) garantissent que des tâches distinctes sur la même ressource ne se chevauchent pas. Malheureusement, un tel programme, compte tenu de la présence de variables entières et binaires potentiellement nombreuses, ne passe pas à l'échelle. Il faut donc envisager une approche non optimale, telle que l'allocation gloutonne, qui a montré de bonnes performances sur des problèmes similaires d'allocation de tâches [1, 22].

Approche gloutonne. L'algorithme glouton, présenté dans l'algorithme 1, trie d'abord les modes par ordre décroissant de récompense (ligne 4). Pour chaque mode, il tente de trouver des créneaux pour toutes les tâches du mode (lignes 5-11). Si toutes les tâches peuvent être programmées, le mode est sélectionné et toutes les tâches sont programmées (lignes 12-14). Sinon, les créneaux réussis sont supprimés du plan des ressources. Pour trouver un créneau, la fonction `first` recherche simplement le premier créneau disponible sur le plan des ressources, compatible avec la fenêtre temporelle de la tâche, sans chevauchement avec une tâche déjà planifiée. Cet algorithme n'est pas optimal, mais fournit des solutions très rapidement, puisqu'il est polynomial en nombre de modes et de tâches. Mais, comme pour le MILP, il nécessite de partager toutes les contraintes et informations avec une autorité centrale de planification. Cet algorithme sera notre référence pour les expérimentations, puisque les algorithmes gloutons sont largement utilisés dans la planification de missions spatiales, par les opérateurs de constellations, et parce que le solveur MILP n'est pas adapté même pour les plus petites instances que nous considérons.

4 Approche décentralisée

Cette section expose la principale contribution de l'article : l'algorithme MM-CBGA basé consensus pour l'allocation de tâches composites à modes multiples. Auparavant, nous rappelons brièvement le contexte de l'allocation de tâches par enchères.

Algorithme 1 : Solveur glouton

Données : Un MACTA $P = \langle \mathcal{A}, \mathcal{R}, \mathring{\mathcal{T}} \rangle$
Résultat : Un plan π

```
1  π ← {}                                          // le plan à construire
2  R ← {(r,[])} | r ∈ R}                           // les plans des ressources
3  S ← {}                                          // les requêtes remplies
4  M ← sort(modes(T̊))                              // les modes triés
5  pour chaque m ∈ M faire
6      si request(m) ∉ S alors                     // requête non remplie
7          T ← {}    // les créneaux trouvés jusqu'à présent
8          pour chaque τ ∈ m faire
9              t ← first(τ,P,R)
10             si t ≠ ∅ alors  // créneau trouvé pour la tâche
11                 T ← T ∪ {t}
12         si |T| = |m|
             alors    // tous les créneaux trouvé pour le mode
13             π ← π ∪ {(τ,t) | t ∈ T}
14             S ← S ∪ {request(m)}
15         sinon       // pas de place pour toutes les tâches
16             remove(T,R)
                // supprimer les créneaux explorés
17 retourner π
```

4.1 Allocation de tâches basée enchères

Le cadre classique d'allocation de tâches consiste en un ensemble de ressources et un ensemble de tâches à exécuter sur ces ressources. L'objectif est d'assigner des tâches aux ressources de manière à optimiser certains critères (e.g., le nombre de tâches assignées, la somme des récompenses des tâches créneaux, la durée totale, etc.). Il s'agit d'un problème d'allocation classique qui peut être modélisé comme un MILP. L'idée est que les tâches à planifier sont ouvertes aux enchères par un *commissaire-priseur*. Les agents *enchérisseurs* évaluent les tâches en fonction de leur plan actuel, et envoient leurs offres pour certaines de ces tâches. Ensuite, le commissaire-priseur détermine les gagnants en fonction de leurs offres et des contraintes sur les ressources. Ici, les opérations les plus coûteuses sont l'étape d'enchérissement effectuée par chaque enchérisseur, qui peut avoir un nombre exponentiel de lots à évaluer, et le problème de détermination du gagnant (WDP) effectué par le commissaire-priseur, qui revient à résoudre un programme linéaire en nombres entiers de taille potentiellement exponentielle, et qui entre dans le cadre des enchères combinatoires (CA) [4].

Si l'on se réfère à la littérature sur l'allocation de tâches multi-robots [5, 11] et sur l'allocation d'observations multi-satellites [18, 13, 19], ces limites de calcul peuvent être surmontées en utilisant la relaxation classique consistant à n'autoriser les enchères que sur des articles individuels (et non sur des lots). Lorsque les enchérisseurs enchérissent sur l'ensemble des tâches en parallèle, nous nous situons dans le cadre du système PSI (*Parallel Single Item*) [12]. Lorsque le commissaire-priseur annonce les tâches une par une, et que les enchérisseurs construisent leur offre en connaissant les allocations de tâches précédentes, nous entrons dans le cadre du *Sequential Single Item* (SSI) [12]. En général, SSI a de très bonnes performances avec un temps de calcul très limité, alors que la qualité des solutions PSI est souvent limitée, puisque les enchérisseurs ne peuvent pas facilement raisonner sur les lots. Plus récemment, l'algorithme d'allocation basé sur le consensus (CBBA) combine les idées des enchères et du consensus pour converger plus rapidement (en tours) que SSI tout en produisant des solutions similaires et en ayant les avantages des algorithmes de consensus traditionnels [2]. CBBA est une solution entièrement distribuée pour mettre en œuvre une variante d'enchères combinatoires à faible coût de calcul. Il suit une séquence répétée en deux phases. Tout d'abord, pendant la *phase d'enchérissement*, chaque agent construit un lot unique de tâches qu'il souhaite se voir attribuer, en respectant le coût marginal associé à l'inclusion de l'élément considéré dans son plan actuel. Ensuite, pendant la *phase de consensus*, les agents comparent leurs offres avec celles de leurs voisins. Si un agent est battu sur une tâche t, il abandonne la tâche et toutes les tâches ajoutées après elle dans le plan, car l'exclusion de t a rendu l'évaluation de leur coût marginal obsolète. Cet algorithme a été largement étudié et modifié pour améliorer ses performances et l'adapter à des scénarios spécifiques, comme l'allocation d'observations multi-satellites [18, 13, 19]. L'extension sur laquelle nous nous concentrons dans cet article est CBGA, *Consensus-based Group Auction* [9]. Ici, certaines tâches multi-agents nécessitent la participation de plusieurs agents pour être réalisées. Lorsqu'un nombre suffisant d'agents enchérissent pour une tâche, celle-ci peut être ajoutée aux lots. Pour ce faire, CBGA diffère de CBBA sur deux points principaux : (i) les structures de données d'offre stockent les agents qui ont fait une offre pour chaque tâche, au lieu de ne stocker que la meilleure valeur d'offre jusqu'à présent, et (ii) pendant le consensus, les récompenses des tâches sont calculées sur la base de la somme des meilleures offres des agents pour effectuer la tâche, en écartant les plus mauvaises lorsque plus d'agents que nécessaire ont fait une offre. Cependant, ni CBBA ni CBGA ne peuvent résoudre notre problème, car MACTA permet que certaines tâches composites soient exécutées de différentes manières (modes). Nous étendons donc CBGA au cadre des tâches composites multi-modes, dans la section suivante.

4.2 MM-CBGA

MM-CBGA suit la même logique que CBBA et CBGA, tel que décrit dans l'algorithme 2. Tout d'abord, les agents résolvent leurs problèmes privés,

Algorithme 2 : Solveur MM-CBGA

Données : Un MACTA $P = \langle \mathcal{A}, \mathcal{R}, \mathring{\mathcal{T}} \rangle$
Résultat : Un plan π^+ pour les requêtes externes, et un plan privé π_i^- pour chaque agent $i \in \mathcal{A}$

1 **pour chaque** $i \in \mathcal{A}$ **faire en parallèle**
2 $\quad \pi_i^- \leftarrow \texttt{solve}(P_i)$
3 $\quad \mathcal{N}_i \leftarrow \{j \in \mathcal{A} \mid j \neq i, \text{can}(j) \cap \text{can}(i) \neq \emptyset\}$
4 **tant que** *conflit* **faire**
5 $\quad$ **pour chaque** $i \in \mathcal{A}$ **faire en parallèle** // Phase de bidding
6 $\quad\quad b_i, c_i \leftarrow \texttt{bid}(i)$ // voir Alg. 3
7 $\quad\quad$ **pour chaque** $j \in \mathcal{N}_i$ **faire** $\texttt{send}(\langle b_i, c_i, s_i \rangle, j)$
8 $\quad$ **pour chaque** $i \in \mathcal{A}$ **faire en parallèle** // Phase de consensus
9 $\quad\quad$ **pour chaque** b_j *reçue* **faire**
10 $\quad\quad\quad \texttt{consensus}(b_i, c_i, b_j, c_j)$ // voir Alg. 4
11 **pour chaque** $i \in \mathcal{A}$ **faire en parallèle** $\texttt{send}(\pi_i^+, \text{client})$
12 **retourner** $\bigcup_{i \in \mathcal{A}} \pi_i^+$

en utilisant n'importe quel algorithme d'ordonnancement (ligne 2) et se connectent aux agents qui peuvent insérer des tâches provenant des mêmes requêtes (ligne 3). Ensuite, chaque agent enchérit simultanément sur l'ensemble des modes, en fonction des offres reçues précédemment, et envoie les résultats à ses voisins (lignes 5-7). Ensuite, chaque agent résout les conflits qu'il peut avoir avec les offres de ses voisins (lignes 8-10). Lorsqu'il n'y a plus de conflit entre les agents, les allocations pour les tâches externes sont renvoyées au client (ligne 11).

Structures de données. Les principales structures de données utilisées par chaque agent i sont les *offres* (ou *bids*) et les *contributions*. Une *offre* est une structure à 3 dimensions où chaque cellule contient la récompense pour une requête donnée $\mathring{\tau}$, un mode donné m et un agent donné j dans $\mathcal{N}_i$, noté $b_i[\mathring{\tau}][m][j]$. La valeur d'une cellule d'offre peut être soit : (i) une valeur positive ou négative finie, lorsque l'agent sait ou évalue combien il gagne ou perd en programmant les tâches du mode dans ses créneaux ; (ii) l'infini négatif, s'il ne peut pas insérer les tâches du mode dans son plan, en raison de la limitation des ressources (plus de place dans ses créneaux) ; (iii) ou vide, s'il n'a pas encore enchéri sur le mode. Les offres sont mises à jour à chaque phase d'enchères, comme expliqué plus loin. Complémentaire à une offre, une *contribution* est l'ensemble des tâches que l'agent j est prêt à programmer pour une requête $\mathring{\tau}$ donnée, et un mode m, noté $c_i[\mathring{\tau}][m][j]$. A chaque offre correspond une contribution, c'est-à-dire l'ensemble des tâches dont l'insertion aboutit à l'offre proposée. Afin d'éviter les blocages dus à des informations obsolètes, chaque agent i stocke la date du dernier message qu'il a reçu de la part de chaque autre agent j, notée $s_i[j]$. Ces horodatages seront également échangés afin d'éviter de prendre des décisions basées sur des informations obsolètes. Enfin, chaque agent i conserve la trace de son *lot* (ou *bundle*), β_i, c-à-d. l'ensemble des modes externes actuellement prévus dans son plan.

Phase d'enchérissement (*bidding*). Contrairement à CBBA et CBGA, où les agents font des offres pour des tâches, l'idée dans MM-CBGA est que les agents font des offres sur les modes pour chaque demande dont ils ont connaissance. L'algorithme 2 présente comment chaque agent met en œuvre cette phase. L'agent tente d'intégrer des modes dans son lot tant que ses créneaux ne sont pas pleins (ligne 4). Pour chaque mode de chaque requête non satisfaite (lignes 6-7), l'agent calcule deux plans -avec et sans le mode-, pour évaluer le coût ou le gain marginal de l'intégration du mode donné dans ses créneaux actuels, en fonction de son lot actuel β_i. Le gain résultant et les tâches respectives constituent l'offre et la contribution de l'agent (lignes 11-12). Les deux plans sont calculés à l'aide d'un algorithme quelconque, noté `solve`, qui peut être MILP ou l'algorithme glouton présentés précédemment. Bien entendu, l'agent n'a pas à recalculer la solution du même sous-problème, en stockant la solution déjà calculée de chaque sous-problème. Le meilleur mode pour chaque demande doit être choisi, mais lorsqu'un agent n'a aucune connaissance des offres des autres agents requises pour un mode, il évaluera une *borne supérieure* pour l'intégration du mode (ligne 13).

$$\texttt{UB}(b_i[\mathring{\tau}][m]) \stackrel{\text{def}}{=} \sum_{b_i[\mathring{\tau}][m][j] \neq \emptyset} b_i[\mathring{\tau}][m][j] \quad (9)$$

$$+ \sum_{\tau \notin \bigcup_j c_i[\mathring{\tau}][m][j]} \omega_\tau \quad (10)$$

Cette borne supérieure consiste en l'agrégation des récompenses effectives des offres existantes, comme dans l'équation (9) et des récompenses hypothétiques maximales pour les offres manquantes sur certaines tâches du mode, comme dans l'équation (10). Dès qu'un agent n'est pas d'accord pour choisir le mode (offre $-\infty$), le mode est écarté pour intégration. La raison pour laquelle on utilise des récompenses hypothétiques et pas seulement des récompenses effectives est de permettre aux modes avec une récompense effective négative d'être considérés, s'ils ont le potentiel d'apporter un certain gain grâce à des récompenses non encore découvertes, comme illustré dans l'exemple 1. Dans le cas où cette borne supérieure est trop confiante, elle sera affinée pendant la phase de consensus, en mettant à jour la récompense avec les offres effectives. Dans le cas où plusieurs agents peuvent effectuer la même tâche pour un mode, et qu'il y a donc plus d'offres sur m pour $\mathring{\tau}$, seules les meilleures offres, suffisantes pour remplir $\mathring{\tau}$, sont utilisées dans la définition de `UB` (les moins bonnes sont ignorées).

Cette borne supérieure est utilisée pour déterminer

Algorithme 3 : La fonction `bid`

```
1  Fonction bid(i)
2      M ← ∅                                   // modes déjà analysés
3      R ← ∅                                   // requêtes déjà analysées
4      tant que true faire
5          ω* ← −∞, τ̊* ← ∅, m* ← ∅
6          pour chaque τ̊ ∈ P̊_i^+ \ R faire          // requêtes
7              pour chaque m ∈ M_τ̊ \ β_i faire      // modes
8                  π_i ← solve(P_i^− ⊎ β_i)
9                  π_i^m ← solve(P_i^− ⊎ (β_i ∪ {m}))
10                 si m ∈ {mode(τ) | (τ,t) ∈ π_i^m} alors
                       // Tâches de m's dans les créneaux de i
11                     c_i[τ̊][m][i] ← {τ | (τ,t) ∈ π_i^m}
                       // évaluer le gain marginal
12                     b_i[τ̊][m][i] ← ω_{π_i^m} − ω_{π_i}
13                     ω ← UB(b_i[τ̊][m])
14                     si ω > ω* alors          // meilleur mode
15                         ω* ← ω, τ̊* ← τ̊, m* ← m
16                 sinon                        // m ne peut être planifié
17                     b_i[τ̊][m][i] ← −∞
18                     c_i[τ̊][m][i] ← ∅
19         si ω* ≠ −∞ alors                     // un mode a été trouvé
20             si m* ⊆ ⋃_j c_i[τ̊*][m*][j]
               alors                            // le mode est plein
21                 si ∑_j b_i[τ̊*][m*][j] > 0 alors
22                     β_i ← β_i ∪ {m*}          // intégrer au lot
23             M ← M ∪ {m*}
               // ne pas considérer m* à nouveau
24             R ← {τ̊ | M_τ̊ ⊆ M}               // ne plus considérer τ̊
25         sinon break
26     retourner b_i, c_i
```

Algorithme 4 : La fonction `consensus`

```
1  Fonction consensus(b_i, c_i, b_j, c_j)
2      conflict ← false
3      pour chaque τ̊ ∈ P̊_i^+ ∩ P̊_j^+ faire // requêtes en commun
4          m_i ← argmax(b_i[τ̊])
5          m_j ← argmax(b_j[τ̊])
6          si b_i[τ̊][m_i] ≠ b_j[τ̊][m_j] alors      // conflit
7              conflict ← true
8              pour chaque m ∈ M_τ̊ faire // mettre à jour
9                  pour chaque k ∈ b_i[τ̊][m][·] faire
10                     si b_j[τ̊][m][k] ≠ ∅ alors
11                         si k ≠ i et s_j[k] > s_i[k] alors
12                             b_i[τ̊][m][k] ←
                                   b_j[τ̊][m][k]
13                             c_i[τ̊][m][k] ←
                                   c_j[τ̊][m][k]
14             m_i* ← argmax_{m∈M_τ̊} {UB(b_i[τ̊][m])}
15             si m_i* ≠ m_i et m_i ∈ β_i alors // nouveau meilleur
16                 discard(β_i, m_i)             // supprimer m_i
```

le prochain meilleur mode candidat à l'intégration dans le lot (lignes 14-15). Une fois que toutes les requêtes et tous les modes restants ont été traités, si un meilleur mode a été trouvé – c-à-d. s'il reste de la place dans les créneaux – et si ce mode est valide, il est inséré dans le lot (lignes 20-22). Un mode est valide lorsque toutes les tâches ont été effectivement choisies (ligne 20), et que sa récompense globale est positive (ligne 21). A la fin de la phase d'enchérissement, chaque agent aura construit son lot, ses offres et ses contributions. Ces deux dernières données sont ensuite envoyées aux agents voisins avec les horodatages, afin d'obtenir un consensus.

Phase de consensus. L'algorithme 4 décrit comment les agents résolvent les conflits pour aligner leurs offres. Lorsque les offres des agents sont en désaccord sur le meilleur mode pour une requête donnée (lignes 4-6), ils résolvent le conflit comme dans CBGA, c-à-d. qu'un agent met à jour ses offres lorsqu'il reçoit des offres de ses voisins avec des informations plus récentes (lignes 8-13). En fonction de ces offres mises à jour et d'un éventuel changement d'avis sur le meilleur mode, un agent peut supprimer des modes de son lot (lignes 14-16). La fonction `discard` supprime du lot un mode et tous les modes insérés après ce mode, afin de reconsidérer l'allocation des tâches.

Convergence. Afin de converger, les algorithmes basés sur le consensus tels que CBBA et CBGA exigent que la récompense de l'insertion d'une tâche dans un ordonnancement ne puisse pas augmenter si d'autres tâches sont insérées avant elle. Ici, cette condition se traduit par : $\texttt{solve}(P_i^- \uplus \beta_i) \leq \texttt{solve}(P_i^- \uplus (\beta_i \cup \{m\}))$. Cette condition, appelée *gain marginal décroissant* [2], est satisfaite dans MM-CBGA puisque nous planifions toujours chaque tâche de chaque mode groupé au moment qui produit l'augmentation maximale du score. Ainsi, si nous essayons d'insérer les mêmes tâches dans un planning plus complet, il y aurait moins de fenêtres de temps disponibles et la tâche devrait être placée soit à un moment moins optimal, soit au même moment que dans le planning original si cette fenêtre de temps ne chevauche toujours pas une autre tâche, comme décrit dans [18].

5 Évaluation expérimentale

Les expérimentations ont été réalisées en Java 11.0.15 et exécutées sur un processeur 20 cœurs Intel(R) Xeon(R) E5-2660 v3 @ 2.60GHz, 62GB RAM, Ubuntu 18.04.5 LTS. Nous évaluons les performances de MM-CBGA en comparaison à deux concurrents de base : un solveur centralisé (centralized, qui est glouton dans notre cas) qui fournit des solutions de bonne qualité (proches de l'optimal) en un temps très limité, et une enchère séquentielle (ssi) qui fournit des solutions de qualité équivalente à CBGA dans des cas mono-mode. La procédure `solve` utilisée dans MM-CBGA et ssi est également l'algorithme glouton présenté dans la sec-

tion 3. Le temps de calcul rapporté ici est un temps de calcul mono-CPU (pas d'exécution distribuée).

5.1 Scénario d'observation de la Terre

Le scénario que nous utilisons pour évaluer MM-CBGA est fortement inspiré de EOSCSP (Earth Observation Satellite Constellation Scheduling Problems) [19], qui consiste initialement en un MACTA multi-mode non-composite (chaque requête est satisfaite par exactement une tâche parmi plusieurs possibles). Ainsi, nous avons généré des instances MACTA pour planifier des tâches d'observation sur un ensemble de satellites avec plusieurs propriétaires de créneaux orbitaux.

Nous considérons une constellation en orbite terrestre basse (500 km d'altitude). La constellation est composée de 4 plans orbitaux avec une inclinaison de 45 degrés. Chaque plan orbital contient deux satellites en opposition de phase, pour un total de 8 satellites. Ce paramètre de constellation est utilisé pour déterminer les propriétaires des créneaux orbitaux et les fenêtres temporelles des tâches en fonction de certains points d'intérêt (POI) sur la Terre, en utilisant une bibliothèque de mécanique spatiale dédiée. Le scénario que nous simulons s'étend sur 6 heures (21600 secondes). Quatre utilisateurs (nos agents, u_1 à u_4) possèdent jusqu'à 10 créneaux orbitaux chacun, et tenteront de répondre aux demandes émises par un cinquième utilisateur (u_0). La propriété des créneaux orbitaux est une fenêtre de visibilité sur des points d'intérêt choisis au hasard (10 parmi 27 villes européennes). Les créneaux sont attribués en utilisant une procédure *round-robin* : chaque utilisateur choisit la meilleure fenêtre de visibilité restante pour observer l'un de ses POIs, et laisse l'utilisateur suivant choisir son prochain créneau d'orbite, jusqu'à ce qu'il n'y ait plus de fenêtre, ou que la limite de 10 créneaux par utilisateur soit atteinte. En moyenne, sur toutes les instances que nous avons considérées, selon l'angle d'incidence maximum ($\theta_{\max} = \frac{\pi}{6}$) pour effectuer des observations de bonne qualité, la durée des créneaux exclusifs est d'environ 10 minutes (≈ 595 secondes).

Tous les utilisateurs émettent des requêtes. Ce nombre varie au cours des expérimentations pour évaluer son impact sur les performances de l'algorithme. Les propriétaires de créneaux orbitaux émettent de 2 à 20 requêtes dans leurs propres créneaux, tandis que l'utilisateur non propriétaire en émet de 4 à 80 (autant que tous les autres utilisateurs) dans des créneaux exclusifs, afin de stresser le système. Une requête est définie par un POI, et donc toutes les fenêtres de visibilité existantes pour acquérir ce POI sont des fenêtres de temps potentielles pour les tâches d'observation. Nous avons étudié deux configurations. La première ne considère qu'un seul mode par requête, qui consiste à faire 5 observations distinctes des mêmes POIs, dans les 5 meilleures fenêtres de visibilité, selon l'angle d'observation. Ainsi, nous nous situons dans la configuration mono-mode des tâches composites, où MM-CBGA est équivalent à CBGA. Dans la deuxième configuration, nous considérons 5 modes par requête, ce qui consiste à dégrader le mode précédemment défini (5 observations) en supprimant 1 à 4 fenêtres. On obtient ainsi 5 modes avec respectivement 5, 4, 3, 2 et 1 tâche par mode. Les fenêtres de temps des tâches sont les mêmes que les fenêtres de visibilité : l'agent peut positionner ses observations n'importe où dans les créneaux de l'orbite. La durée des tâches est déterminée aléatoirement entre 20 et 40 secondes. La planification doit également respecter un temps de reconfiguration inter-tâches des satellites de 10 secondes, puisque les satellites ne peuvent effectuer qu'une seule observation à la fois, et doivent pivoter pour pointer vers le prochain POI. L'angle d'incidence (θ), qui a un impact sur la qualité des observations, est utilisé pour déterminer la récompense de chaque tâche τ, plus il est proche de 0°, mieux c'est : $\omega_\tau = \rho(1 - \frac{|\theta|}{\theta_{\max}})$, avec ρ un nombre aléatoire dans $[1;2]$. On obtient ainsi des récompenses moyennes d'environ 1,41 ($\min \approx 0{,}89$, $\max \approx 1{,}99$).

Nous avons exécuté 100 instances de MACTA générées aléatoirement avec une graine dans $[0:99]$ pour chaque paramètre, et nous avons tracé la moyenne, avec un intervalle de confiance de $[0{,}05;0{,}95]$. Les valeurs aléatoires sont choisies uniformément dans les intervalles fournis. Cela donne un total de 3000 instances.

5.2 Analyse de performance

Pour évaluer les performances de MM-CBGA, nous analysons les métriques de la figure 3, pour deux configurations : 1 mode par requête (ligne du haut) et 5 modes par requête (ligne du bas). Sur chaque ligne, nous représentons le ratio de qualité par rapport au solveur centralisé, le pourcentage de requêtes satisfaites, le nombre de tâches planifiées, le nombre de messages échangés, la charge de communication (en Mo), et le temps de calcul total (en s), avec un nombre croissant de requêtes.

Requêtes mono-mode. Nous examinons tout d'abord les résultats sur des MACTA avec un seul mode par requête. Dans cette situation, les problèmes à résoudre sont équivalents à ceux résolus avec CBGA. Comme prévu, la qualité des solutions de MM-CBGA est équivalente à celle d'une enchère séquentielle (ssi). Puisqu'il n'y a qu'un seul mode par requête, consistant en cinq tâches d'observation, centralized ne peut pas trouver de solutions satisfaisant toutes les requêtes. De plus, avec de nombreuses requêtes (plus de 160), sur ce système à res-

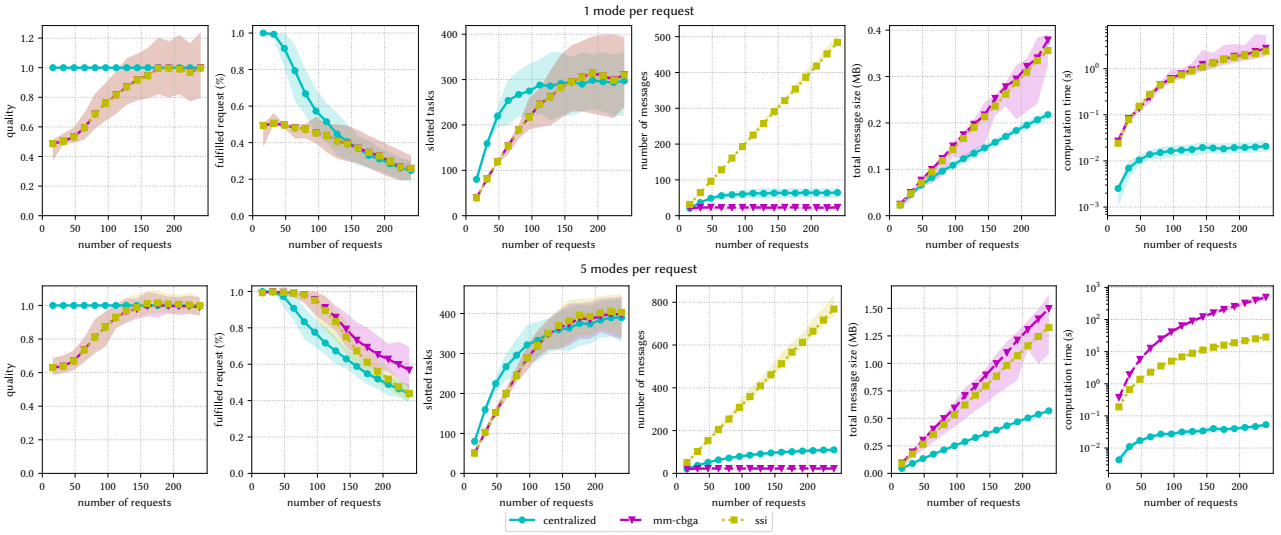

FIGURE 3 – Performances pour les configurations 1 mode par requête (haut) et 5 modes par requête (bas).

sources limitées, centralized est même surclassé par MM-CBGA en termes de requêtes (et donc de tâches) planifiées, car ce dernier permet de planifier des requêtes à récompense non immédiate. Si l'on considère les métriques opérationnelles concernant la communication et le calcul, comme le réseau entre les agents est stable, MM-CBGA ne nécessite qu'un petit nombre constant de messages, alors que ssi nécessite un nombre de messages linéaire par rapport au nombre de modes. Cependant, les structures de données échangées dans MM-CBGA sont quadratiques, ce qui entraîne une charge de communication similaire entre MM-CBGA et ssi. La quantité d'informations est également analysée pour centralized : en effet, les propriétaires de créneaux doivent envoyer toutes leurs données au solveur. En termes de calcul, MM-CBGA, qui nécessite la construction et l'agrégation de données plus importantes (modes fois agents), est plus lent de deux ordres de grandeur que centralized. Ainsi, dans ce cadre mono-mode, MM-CBGA se comporte de manière très similaire à ssi en termes de qualité, de communication et de calcul. En effet, il est équivalent au CBGA. Mais, contrairement à ssi, MM-CBGA est entièrement décentralisé : le WDP est distribué entre tous les agents.

Requêtes multi-mode. Dans cette configuration, chaque requête peut être satisfaite par 5 modes avec une dégradation de 5 tâches à 1 tâche. Ainsi, il y a de la place pour de la relaxation afin de satisfaire plus de requêtes. Ceci est illustré par le pourcentage de requêtes satisfaites, qui ne chute pas aussi rapidement que dans le cadre mono-mode. Il est plus intéressant de noter que MM-CBGA et ssi réalisent plus de requêtes que centralized, pour une qualité moindre sur les petits problèmes (< 150 requêtes). En effet, les deux approches basées sur les enchères remplissent plus de modes avec moins de tâches pour satisfaire plus de requêtes. Sur des instances plus grandes (> 150 requêtes), les courbes de qualité et de satisfaction de ssi et centralized convergent, pour atteindre des performances équivalentes, tandis que MM-CBGA sert encore plus de demandes pour une récompense globale équivalente, par l'utilisation de UB. Sur le plan de la communication, la taille des messages augmente par rapport au paramètre mono-mode pour MM-CBGA, en raison de la présence de modes multiples dans les offres. Pour la même raison, MM-CBGA a besoin de plus de temps pour calculer les offres et résoudre les conflits, ce qui se traduit par un temps de calcul plus élevé (3 ordres de grandeur de plus que centralized, et 1 ordre de grandeur que ssi). Une fois encore, rappelons que MM-CBGA est entièrement décentralisé et que nous présentons un temps mono-CPU. En résumé, dans un cadre multi-mode, MM-CBGA a tendance à répondre à plus de requêtes avec des modes plus petits que centralized, et fournit même des résultats avec une récompense équivalente à centralized sur des instances plus grandes. Le coût de la décentralisation, de la construction incrémentale des lots et de la confidentialité résulte en une qualité inférieure sur des instances plus faciles/petites, et une charge de communication et de calcul plus élevée, en général.

6 Conclusions

Nous avons abordé le problème de l'allocation de requêtes nécessitant la réalisation de plusieurs tâches atomiques et ayant plusieurs modes à réaliser sur des créneaux privés appartenant à des agents coopératifs. Nous avons modélisé ce problème d'allocation (MACTA) et proposé un nouvel algorithme de consensus (MM-CBGA) pour le résoudre de manière décentralisée, en maximisant la récompense globale et sans dévoiler les plans privés.

Nous avons évalué ses performances sur des problèmes de planification de tâches d'observation de la Terre (EOSCSP). MM-CBGA affiche des performances équivalentes à celles des enchères séquentielles (SSI) sur les paramètres mono- et multi-mode, et atteint la même qualité qu'un solveur glouton centralisé sur des instances plus grandes et plus difficiles. En termes de calcul, MM-CBGA nécessite moins d'étapes pour converger mais plus de temps que SSI sur des instances plus grandes. Ce temps de calcul reste inférieur à 5 minutes, mais il est encore possible de l'améliorer en parallélisant certaines opérations. Pour la communication, MM-CBGA nécessite moins de messages, mais plus d'octets, en raison des informations supplémentaires requises pour résoudre les conflits entre les offres. MM-CBGA s'avère être un ajout intéressant à la famille des algorithmes de consensus, comblant le vide en termes de tâches multi-modes et multi-agents.

Nous identifions des améliorations potentielles, notamment en concevant de meilleures heuristiques et des bornes supérieures plus justes, qui pourraient être conçues pour des types spécifiques de tâches ou de modes (e.g., les modes inclus dans d'autres). De plus, il existe d'autres techniques pour coordonner les actions des agents, par exemple les DCOP, à explorer [6]. Ces algorithmes ont été utilisés avec des performances équivalentes à CBBA sur des tâches atomiques multi-mode [19]. Enfin, nous avons considéré des problèmes statiques, alors que les algorithmes de consensus sont adaptés aux problèmes dynamiques, où les demandes et les agents peuvent apparaître ou disparaître. Nous étudierons les performances de MM-CBGA avec des carnets de commande dynamiques en ligne.

Remerciements. Ces travaux ont été menés grâce au financement du gouvernement français dans le contexte du Programme d'Invertissements d'Avenir, au travers du projet BPI PSPC "LiChIE" coordonné par Airbus Defence and Space.

Références

[1] D.-H. Cho, J.-H. Kim, H.-L. Choi et J. Ahn. « Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation ». In : *Journal of Aerospace Information Systems* 15.11 (2018), p. 611-626.

[2] H. Choi, L. Brunet et J. P. How. « Consensus-Based Decentralized Auctions for Robust Task Allocation ». In : *IEEE Trans. Robotics* 25.4 (2009), p. 912-926.

[3] J. Coelho et M. Vanhoucke. « Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers ». In : *European Journal of Operational Research* 213.1 (2011), p. 73-82.

[4] P. Cramton, Y. Shoham et R. Steinberg, éd. *"Combinatorial Auctions"*. MIT Press, 2010.

[5] M. Dias, R. Zlot, N. Kalra et A. Stentz. « Market-Based Multirobot Coordination: A Survey and Analysis ». In : *Proceedings of the IEEE* 94.7 (2006), p. 1257-1270.

[6] F. Fioretto, E. Pontelli et W. Yeoh. « Distributed Constraint Optimization Problems and Applications: A Survey ». In : *Journal of Artificial Intelligence Research* 61 (2018), p. 623-698.

[7] B. P. Gerkey et M. J. Matarić. « A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems ». In : *The International Journal of Robotics Research* 23.9 (2004), p. 939-954.

[8] Gurobi Optimization. *Gurobi Optimizer*. 2022.

[9] S. Hunt, Q. Meng, C. J. Hinde et T. Huang. « A Consensus-Based Grouping Algorithm for Multi-agent Cooperative Task Allocation with Complex Requirements ». In : *Cogn. Comput.* 6.3 (2014), p. 338-350.

[10] IBM Corporation. *IBM ILOG CPLEX Optimization Studio*. 2022.

[11] A. M. Khamis, A. Hussein et A. M. Elmogy. « Multi-robot Task Allocation: A Review of the State-of-the-Art ». In : *Cooperative Robots and Sensor Networks 2015*. T. 604. Studies in Computational Intelligence. Springer, 2015, p. 31-51.

[12] S. Koenig, C. A. Tovey, M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, A. Meyerson et S. Jain. « The Power of Sequential Single-Item Auctions for Agent Coordination ». In : *AAAI 2006*. AAAI Press, 2006, p. 1625-1629.

[13] M. Lee, S. J. Kim, H.-Y. Kim et H.-L. Choi. « Consensus-based Task Scheduling Algorithm for Agile Earth Observation Satellites with Different Authorities ». In : *ASCEND 2021*. 2021.

[14] E. Manisterski, E. David, S. Kraus et N. R. Jennings. « Forming Efficient Agent Groups for Completing Complex Tasks ». In : *AAMAS*. 2006, 834–841.

[15] S. Maqrot, S. Roussel, G. Picard et C. Pralet. « Bundle Allocation with Conflicting Preferences Represented as Weighted Directed Acyclic Graphs – Application to Orbit Slot Ownership ». In : *PAAMS 2022*. T. 13616. LNAI. Springer, 2022, p. 280-293.

[16] S. Maqrot, S. Roussel, G. Picard et C. Pralet. « Orbit Slot Allocation in Earth Observation Constellations ». In : *PAIS 2022*. T. 351. Frontiers in Artificial Intelligence and Applications. IOS Press, 2022, p. 3-16.

[17] N. Michael, M. M. Zavlanos, V. Kumar et G. J. Pappas. « Distributed multi-robot task assignment and formation control ». In : *2008 IEEE International Conference on Robotics and Automation*. 2008, p. 128-133.

[18] S. Phillips et F. Parra. « A Case Study on Auction-Based Task Allocation Algorithms in Multi-Satellite Systems ». In : *AIAA Scitech 2021 Forum*. 2021.

[19] G. Picard. « Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions ». In : *AAMAS*. 2022, p. 1056-1064.

[20] G. Picard. « Multi-Agent Consensus-based Bundle Allocation for Multi-Mode Composite Tasks ». In : *AAMAS*. 2023.

[21] S. Squillaci, S. Roussel et C. Pralet. « Parallel Scheduling of Complex Requests for a Constellation of Earth Observing Satellites ». In : *PAIS 2022*. T. 351. IOS Press, 2022, p. 100-113.

[22] X. Wang, G. Wu, L. Xing et W. Pedrycz. « Agile Earth observation satellite scheduling over 20 years: formulations, methods and future directions ». In : *CoRR* abs/2003.06169 (2020).