Revue Ouverte d'Intelligence Artificielle

Gauthier Picard

Enchères et optimisation multiagent pour la planification de tâches d'observation dans une constellation de satellites

Volume 4, nº 2 (2023), p. 147-168.

https://doi.org/10.5802/roia.60

La Revue Ouverte d'Intelligence Artificielle est membre du Centre Mersenne pour l'édition scientifique ouverte
www.centre-mersenne.org
e-ISSN : 2967-9672

Revue Ouverte d'Intelligence Artificielle
Brouillon d'article, version du 7 septembre 2023

Enchères et optimisation multiagent pour la planification de tâches d'observation dans une constellation de satellites

Gauthier Picard[a]

[a] ONERA/DTIS, Université de Toulouse 2 avenue Édouard Belin 31055 Toulouse CEDEX 4, France
E-mail : gauthier.picard@onera.fr
URL *:* https://www.gauthier-picard.info/.

Résumé. — Nous étudions des techniques distribuées de planification sur des scénarios d'observation de la Terre avec utilisateurs et satellites multiples. Nous nous concentrons sur la coordination des utilisateurs ayant réservé des portions d'orbites exclusives et d'un planificateur central ayant plusieurs demandes qui peuvent utiliser certains intervalles de ces portions exclusives. Nous définissons le problème de planification de constellations de satellites d'observation de la Terre (EOSCSP, ou *Earth Observation Satellite Constellation Scheduling Problem*). Pour le résoudre, nous proposons des schémas multiagents de résolution distribuée, à savoir l'optimisation sous contraintes distribuées et les enchères, dans lesquelles les agents se coordonnent pour répartir les demandes sans partager leurs propres plans. Ces contributions sont évaluées expérimentalement sur des instances EOSCSP générées sur la base de carnets d'observation réels grande échelle ou très conflictuels.

Mots-clés. — Constellation de satellites, planification, allocation de ressources, optimisation distribuée, enchères.

1. Introduction

Ces dernières années ont vu une forte augmentation du développement des constellations de satellites. Au lieu de considérer des satellites individuels, les constellations tirent parti d'un groupe de satellites, dont certains partagent souvent les mêmes plans orbitaux, pour fournir des services plus riches comme le positionnement, les télécommunications ou l'observation de la Terre [17]. Avec peu de satellites dans une constellation (*e.g.* deux dans le projet PLEIADES [9]), et en orbite terrestre basse ou moyenne (altitude inférieure à 35 000 km), toute région de la Terre n'est pas couverte par la constellation à tout moment. Ainsi, la principale motivation pour augmenter la taille de ces constellations est de permettre de capturer avec une grande réactivité n'importe quel point sur Terre, comme le fait la société Planet avec plus de 150 satellites d'observation de la Terre (EOS) [16]. Mais l'exploitation de nombreux EOS nécessite

une meilleure coopération entre les ressources et une autonomie à bord afin d'utiliser au mieux le système, ce qui devient une tâche hautement combinatoire. Outre leur nombre croissant, la composition des constellations évolue également. Les récentes avancées technologiques permettent la production et le déploiement d'EOS agiles capables de changer leur orientation, et de fournir de multiples types de prises de vue avec de multiples capteurs. Tout en offrant des services plus riches à de multiples utilisateurs, cela ajoute de nombreux degrés de liberté et variables de décision pour programmer l'activité des EOS, et ouvre ainsi de nombreux défis [18].

Parmi ces défis, nous nous concentrons sur la planification collective d'observations sur un ensemble de satellites pour lesquels certains utilisateurs ont acquis un *accès exclusif à certaines portions d'orbite* (fenêtres temporelles d'utilisation de satellites), en utilisant des techniques distribuées et multiagents, de manière à répartir les décisions entre les différents utilisateurs de la constellation. Ces utilisateurs exclusifs, bien qu'ayant « acheté » l'exclusivité sur certaines portions d'orbite, ont tout intérêt à intégrer d'autres tâches dans leur plans, si ceux-ci ne sont pas pleins, afin de maximiser l'utilisation de la constellation, et ainsi faire baisser le coût d'exploitation ou être rétribués en proportion des tâches acceptées. La spécificité découlant de la gestion des exclusions et des exigences de confidentialité des tâches programmées dans les fenêtres exclusives entraîne la nécessité de recourir à des méthodes de résolution distribuée. Cette approche répond à de fortes attentes d'utilisateurs souhaitant bénéficier à la fois des avantages d'un système partagé entre plusieurs acteurs (afin de réduire les coûts d'un système global large échelle) et des avantages d'un système propriétaire (afin d'effectuer ce que l'on veut sur sa portion d'orbite, et ceci sans dévoiler ses plans aux autres utilisateurs). Si la littérature sur la planification multi-satellite est riche, comme le confirme un récent article de synthèse [18], considérer les constellations de satellites comme des ressources partagées nécessitant la coordination d'utilisateurs multiples pour la répartition des tâches dans des portions d'orbite exclusives est un problème totalement nouveau, illustré à la figure 1.1, que nous abordons dans cet article.

Au regard de l'existant, des approches basées sur des enchères ont été proposées pour allouer des tâches d'observation à un ensemble de satellites, où chaque satellite est géré par un centre de mission différent [13]. Les centres de mission coordonnent leur allocation à l'aide d'enchères, en mettant aux enchères les observations ouvertes en fonction de leur impact sur le plan à bord et de leur récompense (évaluée à l'aide de l'angle d'incidence des observations programmées). Contrairement à cette approche, dans notre étude, la distribution est justifiée par la présence de certains utilisateurs exclusifs ayant un contrôle total sur certaines portions d'orbite (par *full direct tasking*[(1)]) ou par l'achat ou la réservation long terme de portions d'orbite hors de portée de communication directe. Sur ces fenêtres de temps, il a la priorité absolue pour programmer des observations. Dans un tel contexte, un utilisateur peut demander des

[(1)]Le *full direct tasking* signifie qu'un utilisateur de la constellation peut envoyer directement des tâches à un satellite entrant dans sa fenêtre de communication, puis télécharger les résultats juste avant que le satellite ne soit hors de portée. Cet utilisateur maîtrise le programme sur sa portion d'orbite, mais doit néanmoins respecter certaines contraintes de couplage telles que les contraintes énergétiques, la capacité de mémoire et l'orientation du satellite.

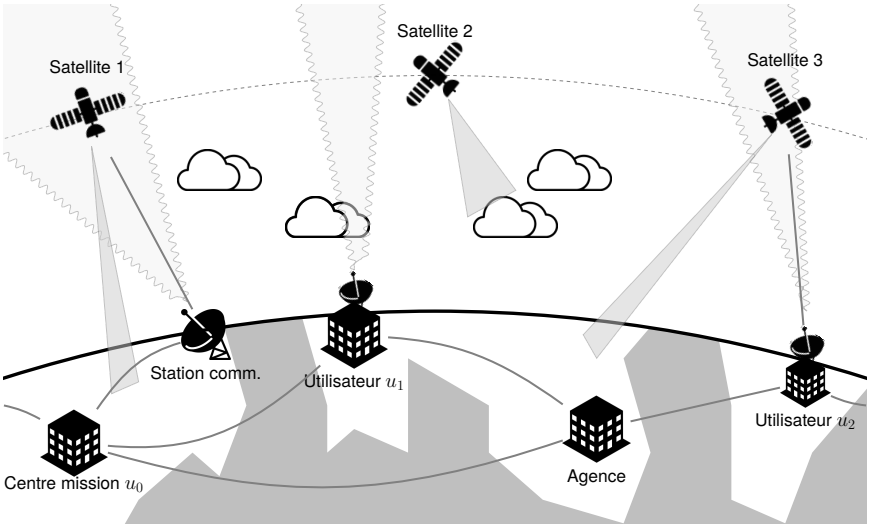

FIGURE 1.1 – Un système EOS composé d'un centre mission principal u_0, des stations réparties sur la surface (avec leur portées de communication), des agences émettant des requêtes d'observation au centre mission, des satellites (avec leur empreinte au sol), et des utilisateurs exclusifs utilisant leurs propres station sol.

observations relativement bon marché sur un satellite par l'intermédiaire d'un planificateur central (centre mission), tout en divulguant ses intentions. Par ailleurs, les utilisateurs peuvent acheter des créneaux horaires exclusifs et utiliser le satellite à titre privé pendant ces périodes. Dans ce cas, le fait que les planifications ne puissent pas être effectuées par une seule autorité, pour des raisons de confidentialité dans des fenêtres exclusives, est une exigence forte. C'est la raison pour laquelle il faut fournir un planificateur distribué où les utilisateurs exclusifs se coordonnent sans divulguer leurs plans, tout en respectant les contraintes de couplage telles que la capacité du satellite ou le temps de configuration entre les observations, qui ne pourraient pas être garanties par des schémas non coordonnés où les utilisateurs font leurs plans en parallèle. Nous étudierons ici deux schémas différents d'allocation et de coordination des ressources distribuées : les enchères et l'optimisation sous contraintes distribuées (DCOP). Dans ces deux approches, les agents seront les utilisateurs exclusifs, devant décider collectivement des requêtes à intégrer dans leurs plans, mais sans les dévoiler. Cet article est une version étendue d'un article présenté aux Journées Francophones sur les Systèmes Multi-Agents [14] qui explorait d'autres schémas de coordination. Ici, nous avons conservé le meilleur candidat basé sur l'optimisation distribuée, et le comparons à des approches basées sur les enchères.

La section 2 illustre et définit le problème de planification de constellation de satellites d'observation de la Terre (EOSCSP). La section 3 se concentre sur les méthodes de résolution centralisées : un programme linéaire et une approche gloutonne

pour EOSCSP. La section 4 présentent des approches par enchères pour résoudre des EOSCSPs, suivant différents schémas (PSI, SSI et CBBA), alors que la section 5 présente une mise en œuvre de la coordination entre les utilisateurs exclusifs en utilisant des techniques d'optimisation distribuée sous contraintes (DCOP). Nous évaluons expérimentalement ces différents algorithmes en utilisant des instances générées de façon aléatoire dans la section 6. Enfin, la section 7 conclut l'article avec quelques perspectives.

2. Le modèle EOSCSP

Cette section illustre le problème que nous étudions à l'aide d'un exemple de scénario, et fournit ensuite quelques définitions de base.

2.1. Scénario illustratif

La figure 2.1 illustre les fenêtres de temps de chaque satellite pour un scénario, où nous considérons : 3 satellites, chacun ayant une période de planification donnée (par exemple, planification sur la prochaine orbite, ou sur les horizons en fonction des fenêtres de communication entre le satellite et les stations au sol) ; 1 utilisateur u_0 sans portion d'orbite exclusive ; 2 utilisateurs ayant des portions d'orbite exclusives telles que u_1 possède des exclusivités (*i.e.* fenêtres temporelles) sur le satellite s_0 et sur le satellite s_1 (rouge hachuré), u_2 possède des exclusivités sur le satellite s_0 et sur le satellite s_2 (bleu hachuré) ; plusieurs requêtes à effectuer avant une date d'échéance, noté $r_{i,j}$ pour la $j$ième requête pour l'utilisateur i ; plusieurs possibilités d'observation par requête, notées $o_{i,j,k}$ pour la $k$ième observation de la $j$ième requête du $i$ième utilisateur. Une seule observation doit être planifiée pour répondre à la requête sur des créneaux temporels en fonction des orbites des satellites et de la position des zones d'intérêt (les créneaux sont représentés sous forme de zones transparentes). Plus précisément, nous considérons 2 observations par requête, de sorte que les observations $o_{1,0,0}$ et $o_{1,0,1}$ sont privées à u_1 (en rouge), les observations $o_{2,0,0}$, $o_{2,0,1}$, $o_{2,1,0}$ et $o_{2,1,1}$ sont privées à u_2 (en bleu), les observations $o_{0,j,k}$ (en vert) sont directement demandées au planificateur central u_0 par d'autres clients sans fenêtre exclusive. La solution de la figure 2.1, représentée par des observations surlignées, répond à toutes les requêtes, en permettant à l'utilisateur non exclusif u_0 de positionner des observations sur des portions d'orbite exclusives (par exemple $o_{0,0,0}$ sur le satellite s_0) . Une contrainte énergétique simplifiée stipule qu'un satellite ne peut pas effectuer plus de $t_{\max}$ minutes d'observation sur sa période de programmation (ici, un maximum de 4 observations est autorisé par satellite), des temps de transition minimaux entre deux observations o et p, en fonction de o et p et de la date à laquelle la transition est déclenchée sur un satellite donné.

Au niveau global, chaque utilisateur exclusif (u_1 ou u_2) peut avoir son propre système de planification pour gérer ses périodes exclusives, et un système de planification central (u_0, l'opérateur de la constellation) gère les observations $o_{0,j,k}$. En fin de compte, chaque utilisateur et le planificateur central ont un problème de planification local à résoudre. Résoudre ces problèmes séparément peut conduire le planificateur

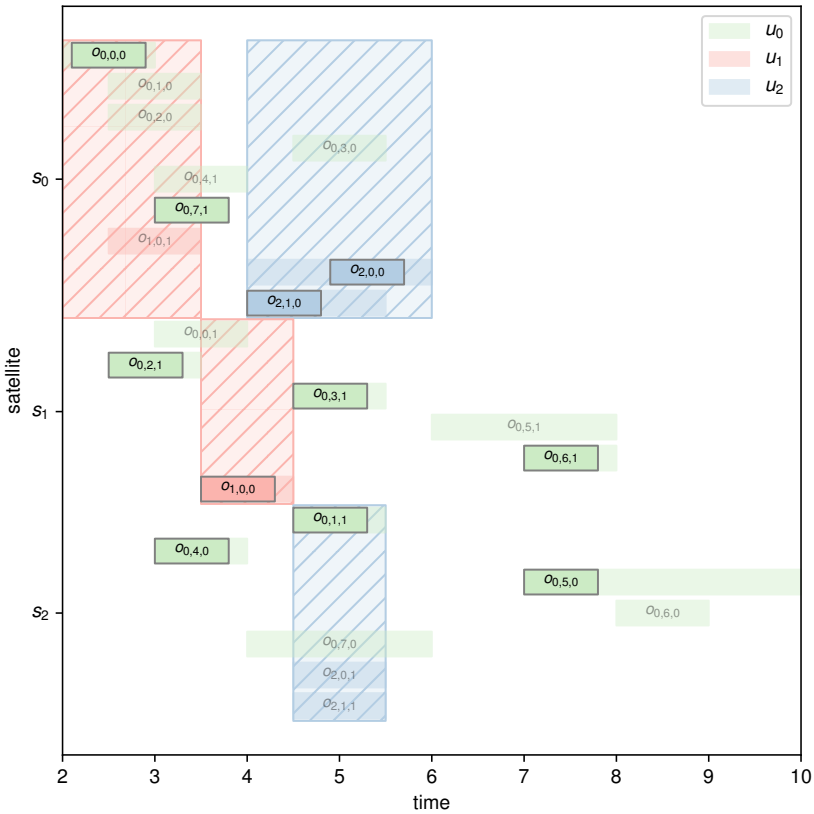

FIGURE 2.1 – Un exemple avec 3 satellites (un par ligne suivant l'axe des ordonnées), 2 utilisateurs (rouge et bleu) avec des exclusives (zones hachées), et 1 utilisateur non exclusif (vert), sur un horizon de temps de 10 pas (axe des abscisses). Les fenêtres de temps d'observation apparaissent comme des surfaces transparentes. Une solution optimale est représentée par des surfaces opaques.

central à ne pas pouvoir réserver de créneaux sur des portions d'orbite exclusives, alors que cela pourrait améliorer la solution. Sans coordination, et avec une gestion non coopérative des créneaux exclusifs, la planification globale pourrait ne pas être optimale, du point de vue du nombre d'observations programmées possibles. Nous proposons donc ici de coordonner les processus de planification entre les utilisateurs.

2.2. DÉFINITIONS ET NOTATIONS

Présentons maintenant les concepts fondamentaux de ce problème de planification.

DÉFINITION 2.1. — *Un* problème de planification de la constellation de satellites d'observation de la Terre avec des exclusivités *(ou EOSCSP) est défini par un tuple* $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, O \rangle$*, tel que* $\mathcal{S}$ *est un ensemble de satellites,* $\mathcal{U}$ *est un ensemble*

d'utilisateurs, $\mathcal{R}$ est un ensemble de requêtes, et $\mathcal{O}$ est un ensemble d'observations à programmer pour répondre aux requêtes de $\mathcal{R}$.

Définition 2.2. — *Un* satellite *est défini comme un tuple $s = \langle t_s^{start}, t_s^{end}, \kappa_s, \tau_s \rangle$ avec $t_s^{start} \in \mathbb{R}$ l'heure de début de son plan d'orbite, $t_s^{end} \in \mathbb{R}$ l'heure de fin de son plan d'orbite, $\kappa_s \in \mathbb{N}^+$ sa capacité (i. e. le nombre maximum d'observations pendant son plan d'orbite), $\tau_s : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$ la fonction définissant le temps de transition entre deux observations données.*

Définition 2.3. — *Un* utilisateur *est défini comme un tuple $u = \langle e_u, p_u \rangle$ avec un ensemble (éventuellement vide) de fenêtres temporelles exclusives*

$$e_u = \{(s, (t^{start}, t^{end})) \mid s \in \mathcal{S}, [t^{start}, t^{end}] \subseteq [t_s^{start}, t_s^{end}])\}\} \subset (\mathcal{S} \times (\mathbb{R} \times \mathbb{R})),$$

et une priorité $p_u \in \mathbb{N}^+$ (utilisée en cas de conflit). On note $\mathcal{U}^{\mathsf{ex}}$ (resp. $\mathcal{U}^{\mathsf{nex}}$) l'ensemble des utilisateurs possédant (resp. ne possédant pas) des exclusivités.

Nous supposons ici qu'un seul utilisateur n'a pas de portion d'orbite exclusive : le planificateur central, noté u_0, c'est-à-dire $\mathcal{U}^{\mathsf{nex}} = \{u_0\}$.

Définition 2.4. — *Une* requête *est définie comme un tuple $r = \langle t_r^{start}, t_r^{end}, \Delta_r, \rho_r, p_r, u_r, \theta_r \rangle$, avec une fenêtre temporelle de validité définie par $t_r^{start} \in \mathbb{R}$ et $t_r^{end} \in \mathbb{R}$, une durée $\Delta_r \in \mathbb{R}$, une récompense $\rho_r \in \mathbb{R}$ si r est réalisée, une position GPS pour observer p_r, un émetteur $u_r \in \mathcal{U}$ et une liste $\theta_r \in 2^{\mathcal{O}}$ d'opportunités d'observation pour valider la requête.*

θ_r est calculée dynamiquement sur la configuration actuelle de la constellation et la position GPS demandée p_r, puisque plusieurs satellites agiles, en changeant leur orientation peuvent acquérir la même position, générant ainsi plusieurs opportunités d'observation.

Définition 2.5. — *Une* observation *est définie comme un tuple $o = \langle t_o^{start}, t_o^{end}, \Delta_o, r_o, \rho_o, s_o, u_o, p_o \rangle$, avec une fenêtre temporelle de validité définie par $t_o^{start} \in \mathbb{R}$ et $t_o^{end} \in \mathbb{R}$, une requête r_o à laquelle elle contribue, une durée $\Delta_o \in \mathbb{R}$, une récompense $\rho_o \in \mathbb{R}$ (héritée de r_o), un satellite s_o sur lequel cette observation peut être planifiée, un émetteur $u_o \in \mathcal{U}$ (hérité de r_o), et une priorité $p_o \in \mathbb{N}^+$ (héritée de r_o).*

Définition 2.6. — *Une* solution *à un EOSCSP est une allocation $\mathcal{M} = \{(o, t) \mid o \in \mathcal{O}, t \in [t_o^{start}, t_o^{end}]\}$ associant une heure de début à au plus une observation par requête de sorte que les utilisateurs exclusifs aient leurs observations planifiées sur leurs fenêtres exclusives respectives, et que la récompense globale soit maximisée (somme des récompenses des observations planifiées) :* $\arg\max_{\mathcal{M}} \sum_{(o,t) \in \mathcal{M}} \rho_o$.

Définition 2.7. — *Un* EOSCSP pour l'utilisateur *u, noté $P[u] = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}[u], \mathcal{O}[u] \rangle$ (ou EOSCSP$[u]$), est un EOSCSP, sous-problème d'un autre EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$ limité aux requêtes et observations appartenant à l'utilisateur u, où $\mathcal{R}[u] = \{r \mid r \in \mathcal{R}, u_r = u\} \subseteq \mathcal{R}$ et $\mathcal{O}[u] = \{o \mid o \in \mathcal{O}, u_o = u\} \subseteq \mathcal{O}$.*

Plus généralement, on note $P[x]$ le problème P limité aux seules composantes liées à x, x étant une requête, une observation ou un satellite. Plus tard, nous utiliserons également les notations $P[\emptyset|\mathcal{M}]$ (resp. $P[u_l, \ldots, u_m|\mathcal{M}]$) pour définir le

problème (resp. sous-problème pour les utilisateurs $u_l, \ldots, u_m$) étant donné une allocation prédéfinie $\mathcal{M}$ de certaines observations. En outre, nous utiliserons la notation $\overline{P}$ pour désigner le EOSCSP P, où seules les requêtes et les observations relatives qui peuvent être planifiées en dehors de toute fenêtre d'exclusivité sont prises en compte (c'est-à-dire les observations dont les fenêtres temporelles croisent des portions d'orbite non exclusives). Enfin, on note l'union de deux problèmes $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$ et $P' = \langle \mathcal{S}', \mathcal{U}', \mathcal{R}', \mathcal{O}' \rangle$, $P \cup P' = \langle \mathcal{S} \cup \mathcal{S}', \mathcal{U} \cup \mathcal{U}', \mathcal{R} \cup \mathcal{R}', \mathcal{O} \cup \mathcal{O}' \rangle$.

3. Approches centralisées à EOSCSP

Nous présentons ici des approches centralisées pour résoudre EOSCSP. Chaque utilisateur ou planificateur a pour objectif de planifier certaines observations sur les satellites. Ce problème de planification des observations peut être modélisé sous la forme d'un programme linéaire en nombres mixtes (MILP). Les variables de décision sont les suivantes. $x_{s,o} \in \{0, 1\}$ est la décision d'effectuer l'observation o sur le satellite s, $t_{s,o} \in \mathbb{R}$ est la date de début de l'observation o sur le satellite s, $\beta_{s,o,p} \in \{0, 1\}$ est la précédence entre deux observations sur le même satellite, qui est égale à 1 si o est avant p sur s, sinon 0.

Nous définissons donc le programme suivant :

$$\max_{x_{s,o}} \quad \sum_{o \in \mathcal{O}, s \in \mathcal{S}} \rho_o x_{s,o} \tag{3.1}$$

t.q.

$$2 - \beta_{s,o,p} - \beta_{s,p,o} \geqslant x_{s,o} \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O} \tag{3.2}$$

$$2 - \beta_{s,o,p} - \beta_{s,p,o} \geqslant x_{s,p} \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O} \tag{3.3}$$

$$\beta_{s,o,p} + \beta_{s,p,o} \leqslant 3 - x_{s,o} - x_{s,p} \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O} \tag{3.4}$$

$$\beta_{s,o,p} + \beta_{s,p,o} \leqslant 1 \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O} \tag{3.5}$$

$$t_{s,p} - t_{s,o} \geqslant \tau_s(o, p) + \Delta_o - \Delta^{\max}_{s,o,p} \beta_{s,o,p}, \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O} \text{ lorsque } \Delta^{\max}_{s,o,p} > 0 \tag{3.6}$$

$$t_{s,o} - t_{s,p} \geqslant \tau_s(p, o) + \Delta_p - \Delta^{\max}_{s,p,o} \beta_{s,p,o}, \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O} \text{ lorsque } \Delta^{\max}_{s,p,o} > 0 \tag{3.7}$$

$$\sum_{o \in \mathcal{O}} x_{s,o} \leqslant \kappa_s \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O} \tag{3.8}$$

$$\sum_{o \in \theta(r)} x_{s,o} \leqslant 1 \qquad \forall s \in \mathcal{S}, \forall r \in \mathcal{R}, \forall o \in \mathcal{O} \tag{3.9}$$

$$x_{s,o} \in \{0, 1\} \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O} \tag{3.10}$$

$$t_{s,o} \in [t_o^{\text{start}}, t_o^{\text{end}}] \subset \mathbb{R} \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O} \tag{3.11}$$

$$\beta_{s,o,p} \in \{0, 1\} \qquad \forall s \in \mathcal{S}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O} \tag{3.12}$$

avec $\Delta^{\max}_{s,o,p} = t_o^{\text{end}} - t_p^{\text{start}} + \Delta_o + \tau^s(o, p)$

(3.2) à (3.7) assurent la précédence des observations et leur séparation par une durée de transition suffisante pour que le satellite change de position. (3.8) fait en sorte que le nombre d'observations planifiées sur un satellite ne dépasse pas sa capacité. (3.9) vérifie qu'au maximum une observation par requête est planifiée. (3.10) à (3.12) sont des définitions de domaines. Ce MILP peut être résolu en utilisant des solveurs standards comme CPLEX ou Gurobi, mais ils ne s'adapteront guère à des problèmes de grande taille (par exemple, plus de 100 observations avec 3 satellites et 3 utilisateurs) car ces configurations génèrent un nombre quadratique de variables entières, et donc des profondeurs d'arbres de recherche trop grandes pour une résolution en temps raisonnable. Pour que les observations des utilisateurs exclusifs aient la priorité sur les observations des utilisateurs non exclusifs, leur récompense doit être fixée à une valeur élevée. Ainsi, le solveur préférera programmer des observations exclusives sur sa fenêtre temporelle plutôt que de planifier une autre observation moins prioritaire. Bien que la solution à ce problème soit optimale, elle exige que chaque utilisateur exclusif *divulgue intégralement les informations sur les requêtes* au planificateur central.

Pour résoudre de plus grands problèmes, une approche consiste à appliquer une allocation gloutonne qui planifie d'abord les observations exclusives des utilisateurs, puis les observations plus urgentes, comme décrit dans l'algorithme 1. En pratique, c'est la technique utilisée par la plupart des opérateurs de satellites/constellations et c'est un étalon classique pour les méthodes de résolution [1, 18]. Pour ce faire, les observations sont triées par ordre croissant selon les critères de priorité et d'heure de début (ligne 2). Ensuite, pour chaque observation de cette liste triée, on trouve le premier créneau libre sur son plan d'orbite de satellite (ligne 4-8), en utilisant la fonction `first_slot`. Cette fonction explore le plan d'un satellite ($R[s]$) afin de trouver le premier créneau libre, respectant les contraintes temporelles de l'observation o. Cet algorithme n'est pas optimal, mais fournit des solutions très rapidement. Cependant, comme pour le MILP, cette solution nécessite de *partager toutes les contraintes et informations* avec un planificateur central.

4. Coordination basée sur des enchères

Une vision pour allouer des ressources et/ou des tâches entre plusieurs agents coopératifs (ici, nos utilisateurs exclusifs) consiste en des approches basées sur les enchères, qui ont prouvé leur flexibilité, leur efficacité, leur équité, et la préservation de la confidentialité des plans et des ressources des utilisateurs. Dans les problèmes d'allocation de tâches multi-robots, de telles approches sont utilisées pour allouer des tâches aux robots, et les intégrer dans leurs plans [4]. Dans notre contexte, on pourrait envisager d'allouer les demandes aux satellites par de tels mécanismes basés sur les enchères, comme proposé dans [13]. Notons que nous nous plaçons bien dans un cadre coopératif, où les utilisateurs exclusifs proposent d'intégrer des tâches dans leur portions d'orbite pour améliorer les bénéfices globaux du système, dont ils sont des partenaires. C'est l'approche que nous suivons dans cette section. Mais d'abord, présentons les mécanismes basés sur les enchères que nous allons mettre en œuvre.

Algorithme 1 : Solveur greedy

Données : Un EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Résultat : Une allocation $\mathcal{M}$

1 $\mathcal{M} \leftarrow \{\}$
2 $\mathcal{O}^{\text{sorted}} \leftarrow \texttt{sort}(\mathcal{O})$
3 $R \leftarrow \{(s, [])\} \mid s \in \mathcal{S}\}$
4 **pour chaque** $o \in \mathcal{O}^{sorted}$ **faire**
5 $\quad t \leftarrow \texttt{first\_slot}(o, P, R)$
6 $\quad$ **si** $t \neq \emptyset$ **alors**
7 $\quad\quad \mathcal{M} \leftarrow \mathcal{M} \cup \{(o, t)\}$
8 $\quad\quad \mathcal{O}^{\text{sorted}} \leftarrow \mathcal{O}^{\text{sorted}} \setminus \theta(r_o)$
9 **retourner** $\mathcal{M}$

10 **Fonction** `first_slot` *(o,* $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$*, R)*
11 $\quad$ **pour chaque** $(s, [t^{start}, t^{end}]) \in \texttt{domains}(o)$ **faire**
12 $\quad\quad$ **si** $|R[s]| < \kappa_s$ **alors**
13 $\quad\quad\quad$ **si** $R[s] = []$ **alors**
14 $\quad\quad\quad\quad$ **si** $t^{end} \geqslant t^{start} + \Delta_o$ **alors**
15 $\quad\quad\quad\quad\quad R[s] = \{(o, (s, t^{\text{start}}))\}$
16 $\quad\quad\quad\quad\quad$ **retourner** (s, t^{start})
17 $\quad\quad\quad$ **sinon**
18 $\quad\quad\quad\quad i \leftarrow 0$
19 $\quad\quad\quad\quad$ **tant que** $i \leqslant |R[s]|$ **faire**
20 $\quad\quad\quad\quad\quad t^{\text{start}\prime} \leftarrow t^{\text{start}}$
21 $\quad\quad\quad\quad\quad$ **si** $i > 0$ **alors**
22 $\quad\quad\quad\quad\quad\quad (o_{i-1}, (s, t_{i-1})) \leftarrow R[s][i-1]$
23 $\quad\quad\quad\quad\quad\quad t^{\text{start}\prime} \leftarrow \max(t^{\text{start}}, t_{i-1} + \Delta_{o_{i-1}} + \tau_{s,o_{i-1},o})$
24 $\quad\quad\quad\quad\quad$ **si** $t^{start\prime} + \Delta_o \leqslant t^{end}$ **alors**
25 $\quad\quad\quad\quad\quad\quad$ **si** $i = |R[s]|$ **alors**
26 $\quad\quad\quad\quad\quad\quad\quad t^{\text{upper}} \leftarrow t^{\text{end}}$
27 $\quad\quad\quad\quad\quad\quad\quad t^{\text{end}\prime} \leftarrow t^{\text{start}\prime} + \Delta_o$
28 $\quad\quad\quad\quad\quad\quad$ **sinon**
29 $\quad\quad\quad\quad\quad\quad\quad (o_i, (s, t_i)) \leftarrow R[s][i]$
30 $\quad\quad\quad\quad\quad\quad\quad t^{\text{upper}} \leftarrow t_i$
31 $\quad\quad\quad\quad\quad\quad\quad t^{\text{end}\prime} \leftarrow t^{\text{start}\prime} + \Delta_o + \tau_{s,o,o_i}$
32 $\quad\quad\quad\quad\quad\quad$ **si** $t^{start\prime} < t^{end\prime} \leqslant t^{upper}$ **alors**
33 $\quad\quad\quad\quad\quad\quad\quad R[s] = \texttt{insert}(R[s], (o, (s, t^{\text{start}\prime})), i)$
34 $\quad\quad\quad\quad\quad\quad\quad$ **retourner** $(s, t^{start\prime})$
35 $\quad\quad\quad\quad\quad i{+}{+}$
36 $\quad$ **retourner** $\emptyset$

4.1. A propos de l'allocation basée sur des enchères

Un cadre générique d'allocation de tâches consiste en un ensemble de ressources et un ensemble de tâches à réaliser par les ressources. L'objectif est d'assigner des tâches aux ressources de manière à maximiser un certain objectif (par exemple, le nombre de tâches assignées ou la somme des récompenses des tâches). Il s'agit donc d'un problème d'allocation classique qui peut être modélisé comme un MILP, comme nous l'avons vu dans la section précédente. Maintenant, l'idée est que les demandes à ordonnancer sont ouvertes à des enchères scellées à un tour par un *commissaire-priseur* (*auctioneer*). Les *enchérisseurs* (*bidders*), qui sont ici les utilisateurs exclusifs, évaluent les demandes en fonction de leur plan actuel, et font une offre pour certaines demandes, comme illustré dans la figure 4.1. Les calculs les plus coûteux dans ce processus sont l'étape des offres (*bidding*) par chaque enchérisseur, qui peut avoir un nombre exponentiel de lots d'articles à évaluer, et le problème de détermination du gagnant (WDP) qui revient à résoudre un programme linéaire en nombre entier avec une taille potentiellement exponentielle, et tombe dans le cadre des enchères combinatoires (CA) [3].

Au regard de la littérature sur l'allocation de tâches multi-robots [4] et l'allocation d'observations multi-satellites [13], pour surmonter ces limites de calcul, la relaxation classique consiste à n'autoriser les enchères que sur un seul article (et non sur des lots), et donc à perdre en optimalité. Lorsque les enchérisseurs enchérissent sur l'ensemble des articles en parallèle, nous nous situons dans le cadre de PSI (*Parallel Single Item*) [8]. Lorsque le commissaire-priseur annonce les articles de manière itérative, et que les enchérisseurs construisent leur offre en connaissant l'allocation précédente des articles, nous nous situons dans le cadre SSI (*Sequential Single Item*) [8]. En général, PSI a de très bonnes performances avec un temps de calcul très limité, mais la qualité des solutions est souvent limitée, car les enchérisseurs ne peuvent pas facilement raisonner sur les lots. Plus récemment, l'algorithme d'enchère par lots basé sur le consensus (CBBA) combine les idées des enchères et du consensus pour converger plus rapidement 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 permettant d'implémenter une variante des enchères combinatoires (CA) à faible coût de calcul. Chaque enchérisseur construit un lot unique d'éléments qu'il souhaite se voir attribuer, en respectant le coût marginal associé à l'inclusion de l'élément considéré dans son lot actuel. Ensuite, pendant la phase de consensus, les enchérisseurs comparent leurs offres avec celles de leurs coéquipiers. Si un agent est surenchéri sur un élément t, il abandonne cet élément et tous les éléments ajoutés après lui, 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 [13] ; à la différence que la distribution n'est pas liée aux satellites, mais aux utilisateurs exclusifs et à leurs portions d'orbite exclusives.

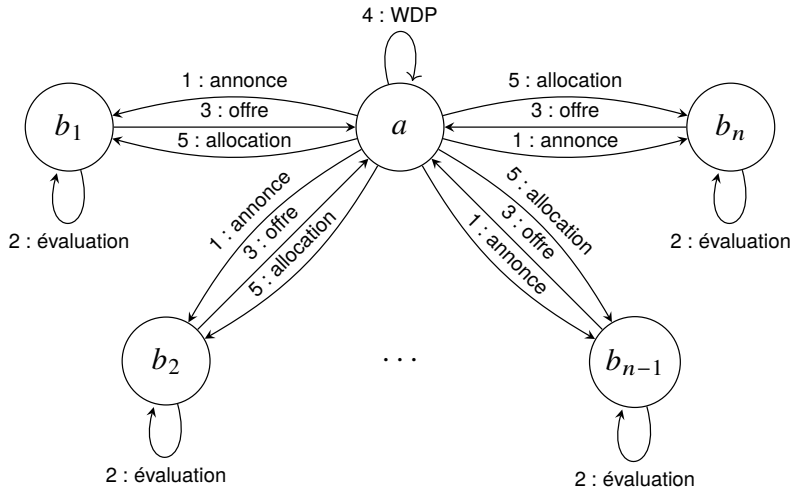

FIGURE 4.1 – Une exemple de processus d'enchère avec un commissaire-priseur a et n enchérisseurs b_i, suivant 5 étapes principales : (1) annonces des articles à allouer, (2) évaluation des articles et des lots par chaque enchérisseur, (3) communication des offres, (4) résolution du problème de détermination des gagnants (WDP), et (5) allocation des articles aux enchérisseurs.

4.2. Résoudre EOSCSP avec des enchères

La mise en correspondance d'un EOSCSP P avec un problème d'allocation basé sur des enchères est assez simple. Les enchérisseurs sont des utilisateurs exclusifs dans $\mathcal{U}^{\text{ex}}$, et les articles sont des requêtes non exclusives dans $\mathcal{R}$ émises par le planificateur central u_0, jouant le rôle de commissaire-priseur. L'idée est que chaque utilisateur exclusif u calcule un plan initial $\mathcal{M}_u$ avec ses propres demandes en résolvant $P[u]$. Ensuite, u_0 annonce les requêtes, soit en bloc (pour PSI et CBBA), soit de manière itérative (pour SSI). Chaque utilisateur exclusif u évalue chaque requête avec la fonction `bid` (pour PSI et SSI) ou chaque lot avec la fonction `bundle` (pour CBBA) en calculant le coût marginal pour intégrer l'élément ou le lot x donné dans son plan actuel. Nous redirigeons le lecteur vers l'article original de CBBA pour plus de détails sur la construction du lot [2]. $\texttt{bid}(r, \mathcal{M}_u)$ revient simplement à résoudre $P[u] \cup P[r]$ et à évaluer la différence avec le plan actuel $\mathcal{M}_u$. Elle renvoie l'offre elle-même $\mathcal{B}_u[r]$ (meilleur coût marginal) et l'ordonnancement d'une observation pour remplir la requête r, $\sigma_u[r] = (o, t)$. Les offres (sur des articles individuels ou des lots) sont ensuite envoyées au commissaire-priseur u_0 (pour PSI et SSI) qui détermine les gagnants, ou aux autres enchérisseurs partageant un intérêt pour la même demande, à savoir $\mathcal{N}_u$, pour trouver un consensus (pour CBBA). Une fois les gagnants déterminés, les demandes sont allouées aux gagnants. S'il reste des demandes non allouées, u_0 tente de les programmer en dehors de toute fenêtre exclusive. Ces processus sont décrits dans les algorithmes 2, 3 et 4.

Algorithme 2 : Solveur psi

Données : Un EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Résultat : Une allocation $\mathcal{M}$

1 $\mathcal{M}_{u_0} \leftarrow \emptyset$
2 **pour chaque** $u \in \mathcal{U}^{ex}$ **faire en parallèle**
3 $\quad \mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
4 $\quad$ **pour chaque** $r \in \mathcal{R}$ **faire** $\mathcal{B}_u[r], \sigma_u[r] \leftarrow \texttt{bid}(r, \mathcal{M}_u)$
$\quad$ `// envoyer` $\mathcal{B}_u, \sigma_u$ `to` u_0
5 **pour chaque** $r \in \mathcal{R}$ **faire**
6 $\quad w \leftarrow \arg\max_{u \in \mathcal{U}^{ex}} \{\mathcal{B}_u[r]\}$
7 $\quad \mathcal{M}_{u_0} \leftarrow \mathcal{M}_{u_0} \cup \{\sigma_w[r]\}$
8 $\quad \mathcal{M}_w \leftarrow \mathcal{M}_w \oplus \sigma_w[r]$ `// envoyer` $\mathcal{M}_w[r]$ `to` w
9 $\mathcal{M}_{u_0} \leftarrow \texttt{solve}(\overline{P[u_0|\mathcal{M}_{u_0}]})$
10 **retourner** $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

Algorithme 3 : Solveur ssi

Données : Une EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Résultat : Une allocation $\mathcal{M}$

1 $\mathcal{M}_{u_0} \leftarrow \emptyset$
2 **pour chaque** $u \in \mathcal{U}^{ex}$ **faire en parallèle** $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
3 **pour chaque** $r \in \texttt{sorted}(\mathcal{R})$ **faire**
4 $\quad$ **pour chaque** $u \in \mathcal{U}^{ex}$ **faire**
5 $\quad\quad \mathcal{B}_u[r], \sigma_u[r] \leftarrow \texttt{bid}(r, \mathcal{M}_u)$
$\quad\quad$ `// send` $\mathcal{B}_u[r], \sigma_u[r]$ `to` u_0
6 $\quad w \leftarrow \arg\max_{u \in \mathcal{U}^{ex}} \{\mathcal{B}_u[r]\}$
7 $\quad \mathcal{M}_{u_0} \leftarrow \mathcal{M}_{u_0} \cup \{\sigma_w[r]\}$
8 $\quad \mathcal{M}_w \leftarrow \mathcal{M}_w \oplus \sigma_w[r]$ `// envoyer` $\mathcal{M}_w[r]$ `to` w
9 $\mathcal{M}_{u_0} \leftarrow \texttt{solve}(\overline{P[u_0|\mathcal{M}_{u_0}]})$
10 **retourner** $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

Dans PSI et SSI, l'opérateur $\oplus$ est utilisé pour ajouter $\sigma_u[r] = (o, t)$ dans le plan actuel. Selon le contexte, il peut s'agir d'une simple agrégation s'il n'y a pas de conflit, ou peut nécessiter de supprimer certaines observations déjà planifiées avec une récompense inférieure. Dans SSI et CBBA, les demandes sont triées avant de boucler. Ce tri peut être fait par rapport à la date d'échéance, la récompense, ou toute combinaison de critères. Dans les expériences de cet article, nous utiliserons la date d'échéance.

Ces approches par enchères permettent ainsi aux utilisateurs exclusifs de se coordonner rapidement afin de décider quelles requêtes (et donc quelles observations) intégrer, sans dévoiler leurs plans privés, contrairement aux approches centralisées.

Algorithme 4 : Solveur cbba

Données : Un EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Résultat : Une allocation $\mathcal{M}$

1 $\mathcal{M}_{u_0} \leftarrow \emptyset$
2 **pour chaque** $u \in \mathcal{U}^{ex}$ **faire en parallèle** $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
3 **pour chaque** $r \in \texttt{sorted}(\mathcal{R})$ **faire**
4 | **pour chaque** $u \in \mathcal{U}^{ex}$ **faire**
5 | | $\mathcal{N}_u \leftarrow \texttt{candidates}(r)$
6 | | $\mathcal{R}_u \leftarrow \mathcal{R}_u \cup \{r\}$
7 **tant que** *conflit* **faire**
8 | **for each** $u \in \mathcal{U}^{ex}$ **do concurrently**
9 | | $\mathcal{B}_u, \mathcal{W}_u, \mathcal{T}_u \leftarrow \texttt{bundle}(u)$
| | `// send` $\mathcal{B}_u, \mathcal{W}_u, \mathcal{T}_u$ `to` $\mathcal{N}_u$
10 | **for each** $u \in \mathcal{U}^{ex}$ **do concurrently**
| | `// résoudre les conflicts et calculer` $\mathcal{M}_u$ `(voir [2])`
11 **pour chaque** $u \in \mathcal{U}^{ex}$ **faire**
12 | $\mathcal{M}_{u_0} \leftarrow \mathcal{M}_{u_0} \cup \{(o,t) | (o,t) \in \mathcal{M}_u, u_o = u_0\}$
13 $\mathcal{M}_{u_0} \leftarrow \texttt{solve}(\overline{P[u_0|\mathcal{M}]})$
14 **retourner** $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

Compte tenu du nombre important de combinaisons possibles, et donc de l'impossibilité d'obtenir une solution optimale en temps raisonnable, nous avons opté pour des approches article par article.

5. Coordination basée sur des DCOPs

Une autre approche pour mettre en œuvre l'allocation des requêtes entre les multiples candidats utilisateurs exclusifs consiste à adopter une vision distribuée d'optimisation sous contraintes. Nous concevons ici un mécanisme de coopération entre utilisateurs exclusifs pour coordonner leur processus d'ordonnancement, en échangeant des messages pour parvenir à un accord sur l'allocation des requêtes tout en respectant les contraintes de couplage, telles que les contraintes de capacité.

5.1. À propos des DCOPs

Une façon de modéliser les problèmes de coordination entre agents consiste à les formaliser dans le cadre des problèmes d'optimisation distribuée sous contraintes (DCOP) [12].

Définition 5.1. — *Un* problème d'optimisation sous contraintes distribuées *discret (ou DCOP) est un tuple* $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$*, où :* $\mathcal{A} = \{a_1, \ldots, a_{|A|}\}$ *est un ensemble d'agents;* $\mathcal{X} = \{x_1, \ldots, x_n\}$ *sont des variables appartenant aux agents;*

$\mathcal{D} = \{\mathcal{D}_{x_1}, \ldots, \mathcal{D}_{x_n}\}$ est un ensemble de domaines finis, tel que la variable x_i prend ses valeurs dans $\mathcal{D}_{x_i} = \{v_1, \ldots, v_k\}$; $C = \{c_1, \ldots, c_m\}$ est un ensemble de contraintes, où chaque c_i définit un coût $\in \mathbb{R}^+ \cup \{+\infty\}$ pour chaque combinaison de l'affectation à un sous-ensemble de variables (une contrainte est initialement connue seulement des agents impliqués) ; $\mu : \mathcal{X} \to \mathcal{A}$ est une fonction associant chaque variable avec l'agent qui la gère ; $f : \prod \mathcal{D}_{x_i} \to \mathbb{R}$ est un objectif représentant le coût global d'une affectation complète de valeurs aux variables.

Les DCOP ont été largement étudiés et appliqués dans de nombreux domaines de référence [7]. Ils ont de nombreuses des propriétés intéressantes, comme : (i) une approche décentralisée où les agents négocient par des échanges locaux de messages ; (ii) une structuration du domaine (en l'encodant dans des contraintes) pour résoudre des problèmes complexes ; (iii) une grande variété de méthodes de résolution allant des méthodes exactes à des techniques heuristiques ou approchées ; comme, par exemple, ADOPT [11], DPOP [12], MaxSum [6], DSA [19] ou MGM [10], pour ne citer que les plus célèbres.

5.2. Coordonner les utilisateurs exclusifs avec des DCOPs

Comme pour les enchères combinatoires, l'utilisation de DCOP pour allouer toutes les requêtes dans leur ensemble est trop coûteuse en calcul pour être mise en œuvre. Nous allons donc nous inspirer de SSI, et considérer les demandes séquentiellement, et coordonner les utilisateurs exclusifs pour chaque demande en utilisant un solveur DCOP, au lieu des enchères. Cela permet aux utilisateurs non exclusifs de se coordonner pour choisir lequel d'entre eux répondra à certaines requêtes en programmant une observation dans ses fenêtres exclusives. Ainsi, pour chaque requête provenant du planificateur central u_0, un nouveau DCOP doit être résolu par l'ensemble des utilisateurs exclusifs intéressés, afin de choisir celui qui planifiera ou non une observation. Comme pour SSI dans le cadres de enchères combinatoires, nous ne considérons les requêtes qu'une à une pour pouvoir résoudre les problèmes en temps raisonnable, au détriment de l'optimalité des solutions.

L'algorithme 5 esquisse cette méthode, appelée `s_dcop` (pour DCOP séquentiel). Tout d'abord, les utilisateurs exclusifs résolvent leur propre sous-problème local de manière parallèle (ligne 1). Ensuite, pour chaque requête r dans la liste ordonnée des requêtes restantes (ligne 2), une nouvelle instance DCOP est construite collectivement entre les utilisateurs exclusifs (ligne 3), puis résolue (ligne 4) en utilisant n'importe quel solveur DCOP disponible (DPOP dans nos expériences). Une fois que toutes les demandes ont été prises en compte, u_0 rassemble les sous-solutions pour construire sa propre solution finale, en programmant autant d'observations en dehors des fenêtres de temps exclusives que possible (lignes 6-7). Notez que le plan interne de chaque utilisateur exclusif reste privé, et que seules les observations non exclusives sont communiquées à u_0 (plus quelques informations supplémentaires pour gérer les temps de transition entre les observations).

Algorithme 5 : Solveur `s_dcop`

Données : Un EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Résultat : Une allocation $\mathcal{M}$

1 **pour chaque** $u \in \mathcal{U}^{\mathsf{ex}}$ **faire** $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
2 **pour chaque** $r \in \texttt{sort}(\mathcal{R})$ **faire**
3 $\quad p \leftarrow \texttt{build\_DCOP}(\theta_r, \mathcal{M}, \mathcal{M}_{u_1}, \ldots, \mathcal{M}_{u_n}, P)$
4 $\quad \mathcal{M}_{u_1}, \ldots, \mathcal{M}_{u_n} \leftarrow \texttt{solve\_DCOP}(p)$
5 $\quad$ **pour chaque** $u \in \mathcal{U}^{\mathsf{ex}}$ **faire**
6 $\quad\quad \mathcal{M}'_u \leftarrow \{(o,t) \in \mathcal{M}_u | u_o \in \mathcal{U}^{\mathsf{nex}}\}$
$\quad\quad$ `// envoyer` $\mathcal{M}'_u$ `to` u_0
7 $\mathcal{M}_{u_0} \leftarrow \texttt{solve}(P[u_0 | \overline{\bigcup_{u \in \mathcal{U}^{\mathsf{ex}}} \mathcal{M}'_u}])$
8 **retourner** $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

5.3. Modèle DCOP

Spécifions maintenant l'instance DCOP à construire à la ligne 3 de l'algorithme 5 pour une requête donnée r, et un plan courant $(\mathcal{M}, \mathcal{M}_{u_1}, \ldots, \mathcal{M}_{u_n})$, comme requis dans la définition 5.1. L'ensemble des agents est l'ensemble des utilisateurs exclusifs qui peuvent potentiellement planifier la requête actuelle r :

$$\mathcal{A} = \{u \in \mathcal{U}^{\mathsf{ex}} | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \exists o \in \theta_r \text{ t.q. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \emptyset\} \quad (5.1)$$

On note $\mathcal{O}[u]^r = \{o \in \theta r | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \text{ t.q. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \emptyset\}$ ces observations liées à la requête r qui peuvent être planifiées sur les fenêtres exclusives u.

La fonction μ associe chaque variable $x_{e,o}$ à son propriétaire e.

Les contraintes doivent vérifier qu'au maximum une observation par requête est planifiée (5.2), que les satellites ne sont pas surchargés (5.3), et qu'au maximum un agent sert la même observation (5.4).

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leqslant 1, \forall u \in \mathcal{A}, \forall o \in \mathcal{O}[u]^r \quad (5.2)$$

$$\sum_{o \in \{o \in \mathcal{O}[u]^r | u \in \mathcal{X}, s_o = s\}, e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leqslant \kappa_s^*, \ \forall s \in \mathcal{S} \quad (5.3)$$

avec κ_s^* la capacité courante du satellite s étant données les observations déjà planifiées $\mathcal{M}, \mathcal{M}_{u_1}, \ldots, \mathcal{M}_{u_n}$.

$$\textstyle\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leqslant 1, \quad \forall o \in \mathcal{O} \quad (5.4)$$

En outre, le coût de l'intégration d'une observation dans l'emploi du temps de l'utilisateur actuel doit être évalué pour guider le processus d'optimisation. Nous

ajoutons donc une contrainte souple à chaque $x_{e,o}$:

$$c(x_{e,o}) = \pi(o, \mathcal{M}_{u_o}), \quad \forall x_{e,o} \in \mathcal{X} \tag{5.5}$$

où π évalue le meilleur coût obtenu lors de la planification de o et toute combinaison d'observations de $\mathcal{M}_{u_o}$, afin de prendre en compte toutes les révisions possibles du plan actuel de u_o. En pratique, au lieu de calculer π à chaque fois, une compilation des contraintes peut être utilisée pour évaluer toutes ces combinaisons une seule fois. En pratique, le coût de chaque alternative parmi cet ensemble de taille exponentielle est calculé à l'aide d'un algorithme glouton polynomial.

Pour résumer, l'ensemble des contraintes du DCOP relatif à la requête r est :

$$C = \{(5.2), (5.3), (5.4), (5.5)\} \tag{5.6}$$

Comme pour les approches par enchères, cette approche par DCOP permet aux utilisateurs exclusifs de se coordonner rapidement afin de décider quelles requêtes (et donc quelles observations) intégrer, sans dévoiler leurs plans privés. Ici encore, pour des raison de temps de calcul raisonnable, nous avons opté pour une approche requête par requête, non optimale.

6. Évaluation expérimentale

Les expérimentations visent à analyser les performances des algorithmes étudiés avec un nombre croissant de requêtes (et donc d'observations). Elles ont été développées en Python 3.7 et exécutées sur un processeur Intel(R) Xeon(R) E5-2660 v3 à 20 cœurs à 2,60 GHz et 62 Go RAM sous Ubuntu 18.04.5 LTS. Nous avons exécuté 30 instances d'EOSCSP générées de manière aléatoire avec une graine dans [0:29] pour chaque taille de problème, et nous avons tracé la moyenne, avec un intervalle de confiance [0,05, 0,95]. La procédure `solve` utilisée dans psi, ssi, cbba et s_dcop est l'algorithme 1, greedy. L'algorithme DCOP utilisé par s_DCOP est l'implémentation de DPOP [12] présente dans pyDCOP [15]. Les valeurs générées de manière aléatoire sont choisies uniformément dans les intervalles fournis.

6.1. Problèmes fortement conflictuels

Nous évaluons les algorithmes sur des problèmes très conflictuels à petite échelle (horizon de planification de 5 min). Cette conflictualité provient du faible nombre de fenêtres exclusives sur lesquelles la plupart des requêtes non exclusives peuvent être positionnées, car chaque requête est associée à 10 opportunités. Les utilisateurs exclusifs auront donc beaucoup de décisions coordonnées à effectuer. Nous générons des EOSCSP avec 3 satellites d'une capacité de 20 observations, 4 utilisateurs exclusifs émettant 2 à 20 requêtes chacun, 8 portions exclusives chacun d'une durée aléatoire dans [15:20], un planificateur central émettant 8 à 80 requêtes, 10 opportunités d'observation par requête d'une durée égale à 5 qui peuvent être planifiées dans une fenêtre de temps d'une durée dans [10:20], et une récompense dans [10:50:10] pour un utilisateur exclusif, et dans [1:5] pour le planificateur central. La fenêtre temporelle des

satellites est [0, 300]. Les temps de transition entre les observations sont uniformément égaux à 1. Les fenêtres exclusives sont positionnées de manière aléatoire, tout en veillant à ce qu'elles ne se chevauchent pas. Les fenêtres temporelles d'observation sont positionnées de manière aléatoire, de manière à garantir qu'elles sont soit incluses dans une fenêtre exclusive, soit en dehors de toute fenêtre exclusive. Il y a beaucoup de chevauchements d'observations, et autant de requêtes provenant du planificateur central que toutes les requêtes des utilisateurs exclusifs.

La figure 6.1 montre les résultats pour ce paramétrage. En termes de récompense, tous les algorithmes distribués sont presque aussi bons que greedy, qui est notre

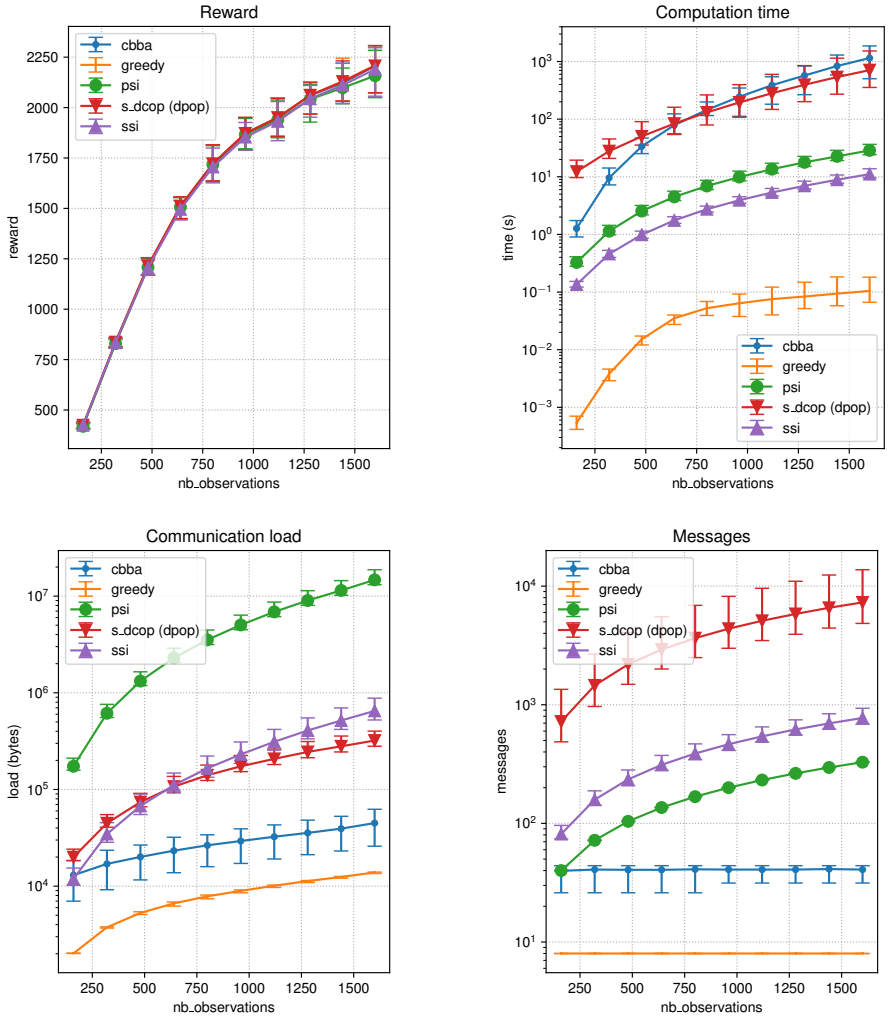

FIGURE 6.1 – Résultats des algorithmes étudiés sur des problèmes fortement conflictuels de petite taille.

référence, car c'est l'algorithme de choix en planification spatiale compte tenu des contraintes de temps de calcul. Néanmoins, cbba et s_dcop fournissent les meilleures solutions distribuées. Le déclin avec le nombre croissant d'observations est dû à la saturation de la capacité des satellites. Les performances de s_dcop et cbba sont au prix d'un temps de calcul supplémentaire, tout en restant raisonnable (environ 1 000 secondes), contrairement au solveur optimal (par exemple CPLEX) qui ne peut pas résoudre les instances avec plus de 100 d'observations (non affiché ici). Le temps de calcul plus élevé de s_dcop résulte du pré-calcul de la fonction π et de la procédure de résolution DPOP sous-jacente. Le surcout de calcul de cbba est dû à l'évaluation des paquets. À un moment donné (problèmes de plus de 750 observations à positionner), cbba nécessite plus de temps de calcul que s_dcop. Cela est dû au nombre exponentiellement croissant de paquets à considérer et au fait qu'à cette taille, avec un tel cadre conflictuel, le réseau de voisinage cbba est un graphe complet, ce qui signifie que chaque utilisateur doit résoudre les conflits avec tous les autres utilisateurs. Du point de vue de la communication, psi échange peu de messages volumineux, puisque toutes les demandes sont communiquées à tous les utilisateurs, ce qui se traduit par un échange de plus de 10 Mo dans les instances plus importantes. ssi et s_dcop échangent de nombreux messages de plus petite taille (en envoyant uniquement des offres sur les requêtes qui les intéressent), en raison du processus séquentiel qu'ils suivent. De son côté, cbba échange moins de messages de petite taille (environ 30 kB au total dans les grandes instances), ce qui en fait un candidat très pertinent dans les environnements distribués, avec un bon compromis entre la qualité de la solution et la charge de communication. Si la réactivité est une exigence, ssi reste le meilleur candidat.

6.2. Problèmes réalistes

Ici, nous générons des EOSCSP de grande taille, avec des paramètres réalistes, dans le respect des carnets de commande fournis par nos partenaires, pour programmer des milliers d'observations dans un horizon de planification de 6 heures. Nous générons des instances comme précédemment mais avec 8 satellites d'une capacité de 500 observations, 5 utilisateurs exclusifs avec 10 à 150 requêtes chacun, 10 portions d'orbite exclusives chacun d'une durée dans [300:600], 1 planificateur central avec 500 à 1 000 requêtes, 5 opportunités d'observation par requête d'une durée égale à 20 qui peuvent être planifiées dans une fenêtre de temps d'une durée dans [40:60]. L'horizon de planification est [0, 21 600]. Cette configuration correspond à la première phase d'une véritable constellation qui sera déployée dans les deux prochaines années.

La figure 6.2 montre les résultats pour ce paramétrage. Tous les algorithmes fournissent des solutions de bonne qualité équivalentes à greedy. Les résultats obtenus dans ce cadre, en se concentrant uniquement sur les observations à l'intérieur de fenêtres exclusives, confirment les performances des différents algorithmes évalués, sauf qu'ici cbba nécessite moins de temps de calcul que s_dcop. Cela est dû au fait que ces instances plus grandes sont moins conflictuelles, et que les voisinages ne sont plus larges. Notons que s_dcop et cbba sont très distribués par nature, et effectuent de nombreux calculs simultanément. Par conséquent, il est possible d'accélérer les

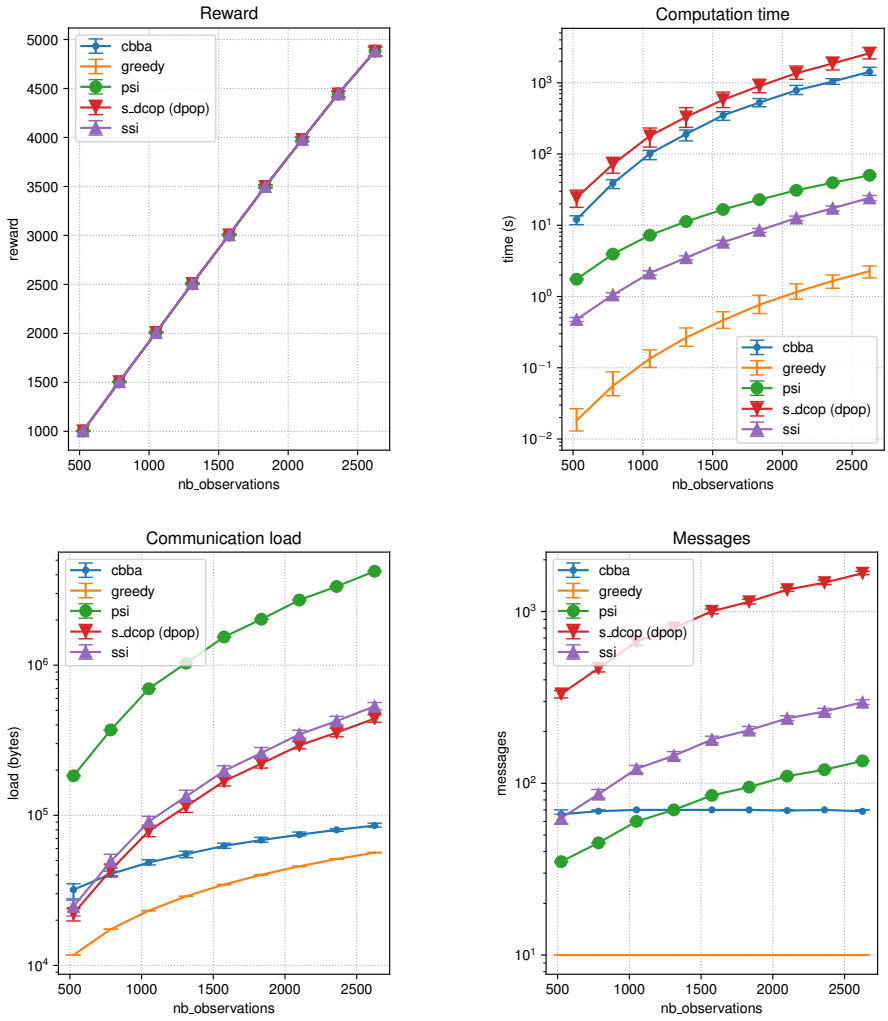

FIGURE 6.2 – Résultats des algorithmes étudiés sur des problèmes réalistes de grande taille.

calculs dans des environnements distribués réels. Tous les algorithmes restent en dessous de 120 minutes de calcul, ce qui permet de les utiliser entre chaque fenêtre de planification (planifier les 6 heures suivantes pendant que le plan actuel de 6 heures est exécuté), même avec un code Python non entièrement optimisé. Concernant les communications, cbba est la meilleure approche pour fournir de bonnes solutions sans divulguer les informations des utilisateurs exclusifs, et nécessite le moins de messages et de quantité de données, pour des instances plus grandes. Ces instances ne sont pas trop conflictuelles, ce qui explique la progression quasi-linéaire de la récompense avec

l'augmentation du nombre de requêtes. Cependant, les carnets de commande des années à venir devraient être de plus en plus conflictuels en raison du nombre croissant de clients de constellations EOS. Dans un tel cas, s_dpop et cbba deviennent encore plus pertinents.

7. Conclusions

Cet article étudie pour la première fois l'utilisation de techniques multiagents distribuées et coopératives, à savoir l'optimisation distribuée sous contraintes et les enchères, pour résoudre un nouveau problème, EOSCSP ; et ce, en respectant la nécessité de limiter la divulgation d'informations entre les utilisateurs. Nous avons défini les composants essentiels d'un EOSCSP et proposé un codage MILP simple pour résoudre de manière optimale de tels problèmes. Malheureusement, ce codage n'est pas utilisable en pratique, même sur de petites instances. Nous avons donc proposé un algorithme glouton et réactif pour résoudre des EOSCSPs. Nous avons conçu et implémenté plusieurs algorithmes distribués (psi, ssi, cbba et s_dcop), qui garantissent de garder les plans internes privés. s_dcop et cbba fournissent des solutions équivalentes aux meilleurs algorithmes évalués sur les problèmes très conflictuels. Cela a un coût : une charge de communication et un temps de calcul plus élevés pour évaluer la récompense d'intégrer une observation dans un planning donné. Cependant, ces techniques sont entièrement distribuables et peuvent bénéficier d'une exécution parallélisée. Sur des problèmes réalistes à grande échelle, la qualité de la solution est encore très bonne par rapport aux greedy. Bien que ces problèmes nécessitent moins de coordination car la probabilité de chevauchement des observations est plus faible, EOSCSP implique toujours de nombreuses observations d'utilisateurs exclusifs, ce qui rend coûteux le calcul de la fonction d'évaluation s_dcop π et la construction des lots dans cbba. Un bon compromis est donc d'utiliser ssi dans des contextes plus larges, puisque le temps de calcul et la charge de communication sont très limités, tout en fournissant des solutions de bonne qualité. Notons que cette investigation a également été un très bon terrain pour confronter les techniques basées sur la DCOP et celles basées sur les enchères, qui ne sont pas souvent comparées dans la littérature. Ces résultats pourraient être généralisés à d'autres problèmes d'allocation de tâches atomiques à des ressources disjointes. Par exemple, EOSCSP pourrait être rapproché d'un problème de type RCPSP (*Resource-Constrained Project Scheduling Problem*) ayant la particularité d'avoir des ressources disjointes de capacité 1 (une seule tâche d'observation à la fois), mais ayant des requêtes multi-modes (un mode par opportunité pour répondre à la même requête) [5].

Ce travail soulève plusieurs perspectives, notamment le développement de solveurs DCOP ou CBBA dédiés et adaptés à la spécificité des EOSCSPs, par exemple l'utilisation de la fonction d'évaluation π ou la construction de lots, qui peuvent résulter d'un processus d'apprentissage, au lieu d'une évaluation systématique de chaque alternative. On peut également envisager de concevoir un langage d'enchères dédié pour évaluer les lots et exécuter le problème de détermination du gagnant de manière efficace. De plus, nous travaillons actuellement à l'intégration des incertitudes (*e.g.*

météo) sur le succès des observations dans le processus de décision, ainsi que l'expression de requêtes composées de plusieurs tâches d'observation élémentaires (*e.g.* périodiques, systématiques) ce qui conduit à des problèmes encore plus complexes à résoudre. Enfin, nous souhaitons pousser les recherches dans la direction de contextes non coopératifs et basés sur le marché, et notamment étudier les incitations pour les utilisateurs exclusifs à accepter des observations dans leurs programmes.

Bibliographie

[1] D.-H. Cho, J.-H. Kim, H.-L. Choi & J. Ahn, « Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation », *Journal of Aerospace Information Systems* **15** (2018), nº 11, p. 611-626.

[2] H. Choi, L. Brunet & J. P. How, « Consensus-Based Decentralized Auctions for Robust Task Allocation », *IEEE Trans. Robotics* **25** (2009), nº 4, p. 912-926.

[3] P. Cramton, Y. Shoham & R. Steinberg (éds.), *Combinatorial Auctions*, MIT Press, 2010.

[4] M. B. Dias, R. Zlot, N. Kalra & A. Stentz, « Market-Based Multirobot Coordination: A Survey and Analysis », *Proceedings of the IEEE* **94** (2006), nº 7, p. 1257-1270.

[5] S. E. Elmaghraby, *Activity Networks: Project Planning and Control by Network Models*, A Wiley-Interscience publication, Wiley, 1977.

[6] A. Farinelli, A. Rogers, A. Petcu & N. R. Jennings, « Decentralised Coordination of Low-power Embedded Devices Using the Max-sum Algorithm », in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, 2008, p. 639-646.

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

[8] S. Koenig, C. A. Tovey, M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, A. Meyerson & S. Jain, « The Power of Sequential Single-Item Auctions for Agent Coordination », in *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, AAAI Press, 2006, p. 1625-1629.

[9] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver & N. Bataille, « Selecting and scheduling observations of agile satellites », *Aerospace Science and Technology* **6** (2002), nº 5, p. 367 - 381.

[10] R. T. Maheswaran, J. P. Pearce & M. Tambe, « Distributed Algorithms for DCOP: A Graphical-Game-Based Approach », in *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems (PDCS)*, 2004, p. 432-439.

[11] P. J. Modi, W. Shen, M. Tambe & M. Yokoo, « ADOPT: Asynchronous distributed constraint optimization with quality guarantees », *Artificial Intelligence Journal* (2005), p. 149-180.

[12] A. Petcu & B. Faltings, « A scalable method for multiagent constraint optimization », in *International Joint Conference on Artificial Intelligence (IJCAI'05)*, 2005, p. 266-271.

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

[14] G. Picard, « Planification multi-utilisateurs et multi-satellites de tâches d'observation dans des constellations avec portions d'orbites exclusives », in *Collectifs cyber-physiques - Vingt-neuvièmes Journées Francophones sur les Systèmes Multi-Agents, Bordeaux, France, June 28-30, 2021* (J. Jamont, éd.), Cépaduès, 2021, p. 117-126.

[15] P. Rust, G. Picard & F. Ramparany, « pyDCOP, a DCOP library for IoT and dynamic systems », in *International Workshop on Optimisation in Multi-Agent Systems (OptMAS@AAMAS 2019)*, 2019.

[16] V. Shah, V. Vittaldev, L. Stepan & C. Foster, « Scheduling the World's Largest Earth-Observing Fleet of Medium-Resolution Imaging Satellites », *IWPSS* (2019), p. 156-161.

[17] J. G. Walker, « Satellite Constellations », *Journal of the British Interplanetary Society* **37** (1984), article no. 559.

[18] X. Wang, G. Wu, L. Xing & W. Pedrycz, « Agile Earth observation satellite scheduling over 20 years: formulations, methods and future directions », `https://arxiv.org/abs/2003.06169`, 2020.

[19] W. Zhang, G. Wang, Z. Xing & L. Wittenburg, « Distributed Stochastic Search and Distributed Breakout: Properties, Comparison and Applications to Constraint Optimization Problems in Sensor Networks », *Artificial Intelligence* **161** (2005), nº 1-2, p. 55-87.

Abstract. — We investigate the use of distributed scheduling techniques on problems related to Earth observation scenarios with multiple users and satellites. We focus on the problem of coordinating users having reserved exclusive orbit portions and one central planner having several requests that may use some intervals of these exclusives. We define this problem as Earth Observation Satellite Constellation Scheduling Problem (EOSCSP) and map it to a Mixed Integer Linear Program. As to solve EOSCSP, we propose multiagent distributed solving schemes, namely Distributed Constraint Optimization and Auctions, where agents coordinate to allocate requests without sharing their own schedules. These contributions are experimentally evaluated on EOSCSP instances based on real large-scale or very conflicting observation order books.

Keywords. — Satellite Constellations, Scheduling, Resource Allocation, Distributed Optimization, Auctions.

Manuscrit reçu le 16 mars 2022, accepté le 22 septembre 2022.