Planification multi-utilisateurs et multi-satellites de tâches d'observation dans des constellations avec portions d'orbites exclusives

Gauthier Picard
gauthier.picard@onera.fr

ONERA-DTIS, Université de Toulouse, France

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 [1]). Pour le résoudre, nous proposons des schémas multiagents de résolution distribués, notamment l'optimisation sous contraintes distribuée, où les agents coopèrent 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, DCOP, allocation de ressources*

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, notably Distributed Constraint Optimization, where agents cooperate 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.

1. Earth Observation Satellite Constellation Scheduling Problem

Keywords: *Satellite constellation, scheduling, DCOP, resource allocation*

1 Introduction

Ces dernières années ont vu une forte augmentation du développement des constellations de satellites. Au lieu de considérer des satellites individuels, elles 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 [20]. Avec peu de satellites dans une constellation (e.g. deux dans le projet PLEIADES [8]), 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) [15]. 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 des variables de décision pour programmer l'activité des EOS, et ouvre de nombreux défis [21].

Parmi ces défis, nous nous concentrons sur la planification collective d'observations sur un ensemble de satellites pour lesquels certains utilisateurs ont un *accès exclusif à certaines por-*

tions d'orbite, en utilisant des techniques distribuées et multiagents, de manière à répartir les décisions entre les différents utilisateurs de la constellation. 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 distribuées. Si la littérature sur la planification multi-satellite est riche, comme le confirme un récent article de synthèse [21] et des travaux récents [1, 12, 18, 16, 2, 3, 6, 17, 22, 15, 11, 21, 7, 19], 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, que nous abordons dans cet article.

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 expose certaines approches distribuées pour résoudre EOSCSP, en utilisant différents schémas de communication entre les agents-utilisateurs, et la section 5 ajoute 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 1 illustre 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 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

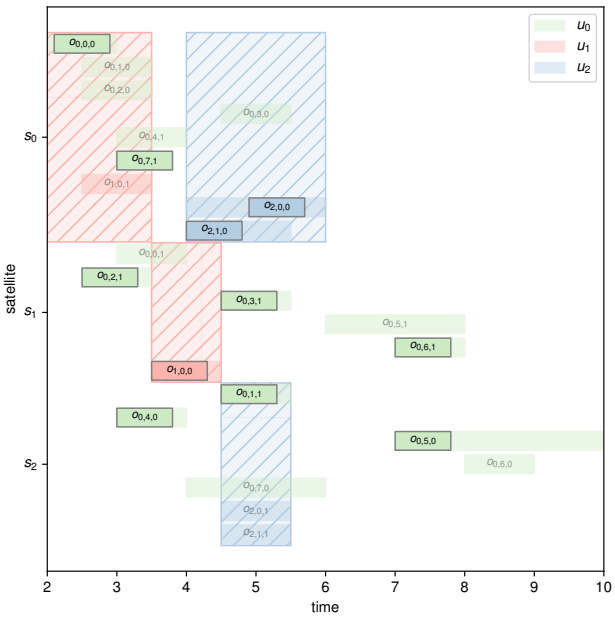

FIGURE 1 – Un exemple avec 3 satellites, 2 utilisateurs (rouge et bleu) avec des exclusives (zones hachées), et 1 utilisateur non exclusif (vert). Les fenêtres de temps d'observation apparaissent comme des surfaces transparentes. Une solution optimale est représentée par des surfaces opaques.

$j$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 personnelles 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 personnelles 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 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 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 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 $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$, 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. Un *satellite* est défini comme un tuple $s = \langle t_s^{\text{start}}, t_s^{\text{end}}, \kappa_s, \tau_s \rangle$ avec $t_s^{\text{start}} \in \mathbb{R}$ l'heure de début de son plan d'orbite, $t_s^{\text{end}} \in \mathbb{R}$ 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 3. Un *utilisateur* est défini comme un tuple $u = \langle e_u, p_u \rangle$ avec un ensemble (éventuellement vide) de fenêtres temporelles exclusives $e_u = \{(s, (t^{\text{start}}, t^{\text{end}})) \mid s \in \mathcal{S}, [t^{\text{start}}, t^{\text{end}}] \subseteq [t_s^{\text{start}}, t_s^{\text{end}}])\} \subset (\mathcal{S} \times (\mathbb{R} \times \mathbb{R}))$, et une priorité $p_u \in \mathbb{N}^+$ (utilisée en cas de conflit). On note $\mathcal{U}^{\text{ex}}$ (resp. $\mathcal{U}^{\text{nex}}$) 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 $\mathcal{U}^{\text{nex}} = \{u_0\}$.

Définition 4. Une *requête* est définie comme un tuple $r = \langle t_r^{\text{start}}, t_r^{\text{end}}, \Delta_r, \rho_r, p_r, u_r, \theta_r \rangle$, avec une fenêtre temporelle de validité définie par $t_r^{\text{start}} \in \mathbb{R}$ et $t_r^{\text{end}} \in \mathbb{R}$, 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 5. Une *observation* est définie comme un tuple $o = \langle t_o^{\text{start}}, t_o^{\text{end}}, \Delta_o, r_o, \rho_o, s_o, u_o, p_o \rangle$, avec une fenêtre temporelle de validité définie par $t_o^{\text{start}} \in \mathbb{R}$ et $t_o^{\text{end}} \in \mathbb{R}$, 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 6. Une *solution* à un EOSCSP est une allocation $\mathcal{M} = \{(o, t) \mid o \in \mathcal{O}, t \in [t_o^{\text{start}}, t_o^{\text{end}}]\}$ 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 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 :

$$\max_{x_{s,o}} \quad \sum_{o \in \mathcal{O}, s \in \mathcal{S}} \rho_o x_{s,o} \tag{1}$$

t.q. $\forall s \in \mathcal{S}, \forall r \in \mathcal{R}, \forall o \in \mathcal{O}, \forall p \in \mathcal{O}$

$$2 - \beta_{s,o,p} - \beta_{s,p,o} \geq x_{s,o} \tag{2}$$

$$2 - \beta_{s,o,p} - \beta_{s,p,o} \geq x_{s,p} \tag{3}$$

$$\beta_{s,o,p} + \beta_{s,p,o} \leq 3 - x_{s,o} - x_{s,p} \tag{4}$$

$$\beta_{s,o,p} + \beta_{s,p,o} \leq 1 \tag{5}$$

$$t_{s,p} - t_{s,o} \geq \tau_s(o,p) + \Delta_o - \Delta^{\max}_{s,o,p} \beta_{s,o,p}, \quad \Delta^{\max}_{s,o,p} > 0 \tag{6}$$

$$t_{s,o} - t_{s,p} \geq \tau_s(p,o) + \Delta_p - \Delta^{\max}_{s,p,o} \beta_{s,p,o}, \quad \Delta^{\max}_{s,p,o} > 0 \tag{7}$$

$$\sum_{o \in \mathcal{O}} x_{s,o} \leq \kappa_s \tag{8}$$

$$\sum_{o \in \theta(r)} x_{s,o} \leq 1 \tag{9}$$

$$x_{s,o} \in \{0,1\} \tag{10}$$

$$t_{s,o} \in [t^{\text{start}}_o, t^{\text{end}}_o] \subset \mathbb{R} \tag{11}$$

$$\beta_{s,o,p} \in \{0,1\} \tag{12}$$

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

(2) à (7) assurent la précédence des observations et que leur distance est au moins le temps de transition nécessaire sur leur satellite. (8) fait en sorte que le nombre d'observations planifiées sur un satellite ne dépasse pas sa capacité. (9) vérifie qu'au maximum une observation par requête est planifiée. (10) à (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). 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 [3, 21]. 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). 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.

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 $t \leftarrow \texttt{first\_slot}(o, P, R)$
6 **si** $t \neq \emptyset$ **alors**
7 $\mathcal{M} \leftarrow \mathcal{M} \cup \{(o,t)\}$
8 $\mathcal{O}^{\text{sorted}} \leftarrow \mathcal{O}^{\text{sorted}} \setminus \theta(r_o)$
9 **retourner** S

4 Approches distribuées à EOSCSP

Nous étudions ici les schémas de coordination pour résoudre le problème EOSCSP de manière collective et discutons de la mesure dans laquelle les informations sur les utilisateurs doivent être divulguées pour résoudre le problème EOSCSP.

4.1 Résolution simple par communication d'exclusifs à non-exclusif

Une approche simple pour envisager de résoudre EOSCSP de manière distribuée consiste à mettre en œuvre une version distribuée de l'algorithme glouton susmentionné, où chaque utilisateur exclusif planifie ses observations privées dans ses exclusions, puis le planificateur central rassemble les observations planifiées afin de positionner les observations restantes demandées par les autres utilisateurs. Cette approche est distribuée et rapide, mais les observations des utilisateurs exclusifs doivent être communiqués au planificateur central afin de construire le plan collectif final. L'algorithme 2 esquisse ce processus de résolution distribué, appelé ex2nex. La fonction `solve` est un raccourci vers n'importe quel solveur EOSCSP (par exemple, basé sur le MILP ou glouton) utilisé d'abord par chaque utilisateur exclusif pour obtenir une solution locale (ligne 1), et ensuite par le planificateur central u_0 pour positionner les observations restante avec la connaissance des solutions locales (ligne 2).

Algorithme 2 : Solveur ex2nex

Données : An EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Résultat : An assignment $\mathcal{M}$
1 **pour chaque** $u \in \mathcal{U}^{\text{ex}}$ **faire en parallèle**
  $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
2 **retourner** $\texttt{solve}(P[u_0 | \bigcup_{u \in \mathcal{U}^{\text{ex}}} \mathcal{M}_u])$

4.2 Résolution simple par communication non-exclusif à exclusifs

Afin de limiter la divulgation des plans des utilisateurs exclusifs, une approche symétrique à ex2nex consiste à considérer que le planificateur central planifie autant d'observations que possible en dehors des fenêtres exclusives, puis à demander aux utilisateurs exclusifs de planifier les observations restantes dans leurs fenêtres respectives (conjointement avec leurs propres observations). Ainsi, les utilisateurs exclusifs ne partagent pas leurs plans, seules les requêtes et observations non exclusives sont partagées avec les utilisateurs exclusifs intéressés. L'algorithme 3 esquisse cette approche, appelée nex2ex. Tout d'abord, le planificateur central positionne les observations non exclusives en dehors des fenêtres exclusives (ligne 1), puis chaque utilisateur exclusif planifie ses propres observations et les observations non exclusives restantes, en fonction des observations déjà planifiées (lignes 2-3). Enfin, le planificateur central rassemble toutes les observations non exclusives planifiées (lignes 4-5). nex2ex a l'avantage de *de sécuriser les informations sur les horaires des utilisateurs exclusifs*. Seules les observations des utilisateurs non exclusifs dont les fenêtres temporelles chevauchent des portions d'orbite exclusives sont partagées par le solveur central. Toutefois, cela peut conduire à des *requêtes surbookées*, c'est-à-dire des requêtes avec plus d'une observation planifiée sur différentes fenêtres exclusives. De plus, la contrainte de respect de la capacité des satellites n'est pas garantie.

Algorithme 3 : Solveur nex2ex

Données : Un EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Résultat : Une allocation $\mathcal{M}$
1 $\mathcal{M} \leftarrow \texttt{solve}(P[u_0])$
2 **pour chaque** $u \in \mathcal{U}^{\text{ex}}$ **faire en parallèle**
3 $\mathcal{M}_u \leftarrow \texttt{solve}(P[u, u_0 | \mathcal{M}])$
4 $\mathcal{M}'_u \leftarrow \{(o,t) \in \mathcal{M}_u | u_o \in \mathcal{U}^{\text{nex}}\}$
  `// send` $\mathcal{M}'_u$ `to` u_0
5 **retourner** $\mathcal{M} \cup \bigcup_{u \in \mathcal{U}^{\text{ex}}} \mathcal{M}_u$

4.3 Résolution itérative

Pour éviter la surréservation de l'approche nex2ex, on peut considérer l'allocation comme un processus itératif, où les requêtes et les observations ne sont pas considérées par lots, mais une par une et envoyées à certains utilisateurs candidats. Cette approche, que nous appelons it-nex2ex, est décrite dans l'algorithme 4.

Comme pour nex2ex, une première solution est obtenue pour les observations qui peuvent être planifiées en dehors des fenêtres exclusives (ligne 1). Ensuite, les observations restantes sont triées, selon l'heure de début, la priorité ou tout autre critère (ligne 2). Cette liste triée est ensuite parcourue, et pour chaque observation, des utilisateurs exclusifs candidats sont choisis (ligne 4). Ce choix peut être arbitraire ou basé sur certains indicateurs comme la récompense ou la probabilité que l'observation soit de bonne qualité (si un

Algorithme 4 : Solveur itnex2ex

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

1 $\mathcal{M} \leftarrow \texttt{solve}(P[u_0])$
2 $\mathcal{O}^{\text{sorted}} \leftarrow \texttt{sort}(\mathcal{O} \setminus \{o | (o,t) \in \mathcal{M}\})$
3 **pour chaque** $o \in \mathcal{O}^{sorted}$ **faire**
4 **pour chaque** $u \in \texttt{candidates}(o, \mathcal{U}^{ex})$ **faire**
5 $\mathcal{M}_u \leftarrow \texttt{solve}(P[u, u_0 | \mathcal{M}] \cup P[o])$
6 $\mathcal{M}'_u \leftarrow \{(o,t) \in \mathcal{M}_u | u_o \in \mathcal{U}^{\text{nex}}\}$
  // envoyer $\mathcal{M}'_u$ à u_0
7 **retourner** $\mathcal{M} \cup \bigcup_{u \in \mathcal{U}^{ex}} \mathcal{M}'_u$

tel modèle existe ou peut être appris). Dans cet article, nous allons considérer l'ordre lexicographique arbitraire. Chaque candidat essaie ensuite d'intégrer l'observation dans son plan (ligne 5), et envoie le résultat au planificateur central (ligne 6). Pour améliorer les performances, cela peut se faire en deux étapes : (1) essayer d'ajouter o de manière gloutonne et conservatrice, sans déplacer les observations déjà planifiées de u ; (2) réviser le plan de u pour intégrer o au cas où la première étape échouerait. Cela évite de recalculer le plan de u à chaque o interrogé. Enfin, la solution est construite en agrégeant les sous-solutions provenant d'utilisateurs exclusifs. itnex2ex *empêche la surréservation* par rapport à nex2ex en considérant itérativement certains utilisateurs candidats. La divulgation d'informations est équivalente à nex2ex, mais le calcul est synchrone et ne bénéficie pas de la distribution. Encore une fois, la contrainte de respect de la capacité des satellites n'est pas garantie.

5 Coordination par DCOP

Afin d'améliorer itnex2ex, nous étudions une approche orientée multi-agents et concevons un mécanisme de coopération entre utilisateurs exclusifs pour coordonner la planification.

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) [13].

Définition 8. 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 $\mathcal{D}_{x_i} = \{v_1, \ldots, v_k\}$; $\mathcal{C} = \{c_1, \ldots, c_m\}$ 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) ; $\mu : \mathcal{X} \rightarrow \mathcal{A}$ est une fonction associant chaque variable avec l'agent qui la gère ; $f : \prod \mathcal{D}_{x_i} \rightarrow \mathbb{R}$ est un objectif représentant le coût global d'une affectation complète de valeurs aux variables. L'objectif d'optimisation est représenté par la fonction f, qui, en général, est considérée comme la somme des coûts : $f = \sum_i c_i$. Une *solution* à un DCOP P est une affectation complète à toutes variables. Une solution est *optimale* si elle minimise f.

Les DCOP ont été largement étudiés et appliqués dans de nombreux domaines de référence [5]. 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 [10], DPOP [13], MaxSum [4], DSA [23] ou MGM [9], pour ne citer que les plus célèbres.

5.2 Étendre itnex2ex avec des DCOPs

Le problème avec itnex2ex est que la prise en compte itérative des observations (i) nécessite de choisir les candidats et leur ordre, et (ii) empêche d'exploiter la distribution de certains calculs. Nous allons donc examiner ici les requêtes de manière itérative, et laisser les utilisateurs exclusifs se coordonner pour choisir celui qui y répondra en planifiant une observation dans ses fenêtres temporelles 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. L'algorithme 5 esquisse cette méthode, appelée itnex2ex_DCOP. Tout d'abord, le planificateur central tente de positionner autant d'observations que possible en dehors des fenêtres temporelles exclusives (ligne 1). Les utilisateurs exclusifs résolvent également leur propre sous-problème local simultanément (ligne 2).

Ensuite, pour chaque requête r dans la liste ordonnée des requêtes restantes (ligne 3-4), une nouvelle instance DCOP est construite collectivement entre les utilisateurs exclusifs (ligne 5), puis résolue (ligne 6) en utilisant n'importe quel solveur DCOP disponible. Une fois que toutes les requêtes ont été examinées, le planificateur central rassemble les sous-solutions pour construire sa propre solution finale (ligne 7-8).

Algorithme 5 : Solveur itnex2ex_DCOP

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

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

5.3 Modèle DCOP

Spécifions maintenant l'instance DCOP à construire à la ligne 5 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 8. L'ensemble des agents est l'ensemble des utilisateurs exclusifs qui peuvent potentiellement planifier la requête actuelle r :

$$\mathcal{A} = \{u \in \mathcal{U}^{\text{ex}} | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \exists o \in \theta_r \text{ t.q. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \emptyset\} \quad (13)$$

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 (14), que les satellites ne sont pas surchargés (15), et qu'au maximum un agent sert la même observation (16).

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \forall u \in \mathcal{A}, \forall o \in \mathcal{O}[u]^r \quad (14)$$

$$\sum_{o \in \{o \in \mathcal{O}[u]^r | u \in \mathcal{X}, s_o = s\}, e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq \kappa_s^*, \ \forall s \in \mathcal{S} \quad (15)$$

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

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \quad \forall o \in \mathcal{O} \quad (16)$$

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 $x_{e,o}$:

$$c(x_{e,o}) = \pi(o, \mathcal{M}_{u_o}), \quad \forall x_{e,o} \in \mathcal{X} \quad (17)$$

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 :

$$\mathcal{C} = \{(14), (15), (16), (17)\} \quad (18)$$

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 ex2nex, nex2ex, itnex2ex et itnex2ex_DCOP est l'algorithme glouton greedy. L'algorithme DCOP utilisé par itnex2ex_DCOP est l'implémentation de DPOP [13] présente dans pyDCOP [14]. 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 lesquelle la plupart des requêtes non exclusives peuvent être positionnées, car chaque requête est associée à 10 ooportunités. Les utlisateurs 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 2 montre les résultats pour cette configuration. En ce qui concerne les récompenses, itnex2ex_dpop offre des performances légèrement en-deça (environ 5.6%) des approches sans privacité (greedy et ex2nex). Les autres algorithmes avec privacité (nex2ex et itnex2ex) affichent des performances se dégradant avec une augmentation du nombre de requêtes. Ceci est dû au manque de coordination qui implique une augmentation du nombre de surbookings et de surcharge des satellites. Lorsque de tels cas surviennent, les observations en question sont retirées du plan, et donc les récompenses sont réduites. Les performances de itnex2ex_dpop sont au prix d'un temps de calcul supplémentaire, tout en restant raisonnable, contrairement à un solveur MILP (par exemple, CPLEX) qui ne peut pas résoudre les instances avec plus de 100 observations (non affiché ici). Le temps de calcul plus élevé de itnex2ex_dpop résulte de la fonction de pré-calcul π et de la procédure de résolution DPOP sous-jacente. itnex2ex_dpop génère une charge de communication supplémentaire (environ 100kB pour les instances les plus grandes) en raison du processus itératif et de l'échange de messages DPOP. En résumé, itnex2ex_dpop est un candidat pertinent qui fournit de bonnes solutions sans divulguer d'informations sur les utilisateurs exclusifs.

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 1000 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, 21600].

La figure 3 montre les résultats pour ce paramètrage. Tous les algorithmes fournissent des solutions de bonne qualité, équivalentes à greedy, sauf nex2ex qui est environ 1,5% en dessous. ex2nex fournit même des résultats légèrement plus rapidement que greedy. Au niveau du temps de calcul, tous les algorithmes restent en dessous de 180 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. Dans ce contexte, itnex2ex_dpop est la meilleure approche pour fournir de bonnes solutions sans divulguer les informations des utilisateurs exclusifs, mais nécessite l'échange de 30 Mo 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

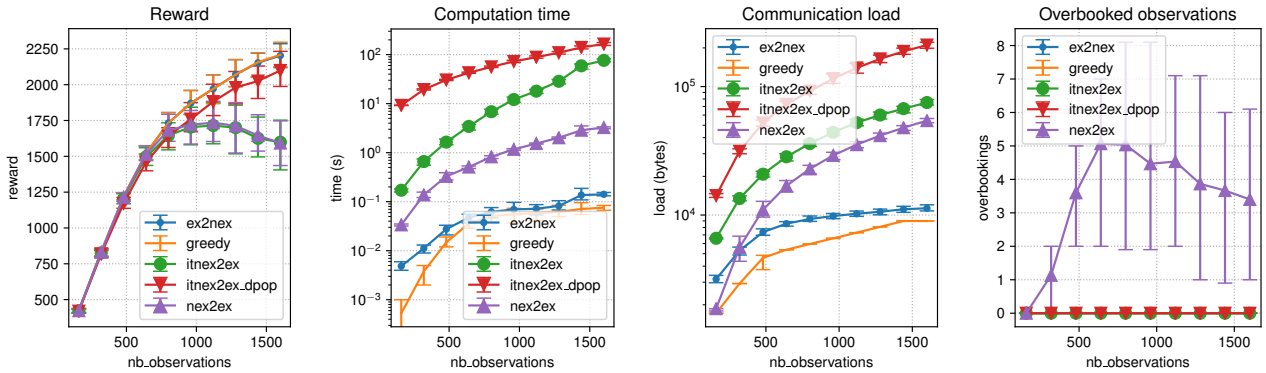

FIGURE 2 – Résultats des algorithmes étudiés sur des problèmes fortement conflictuels de petite taille.

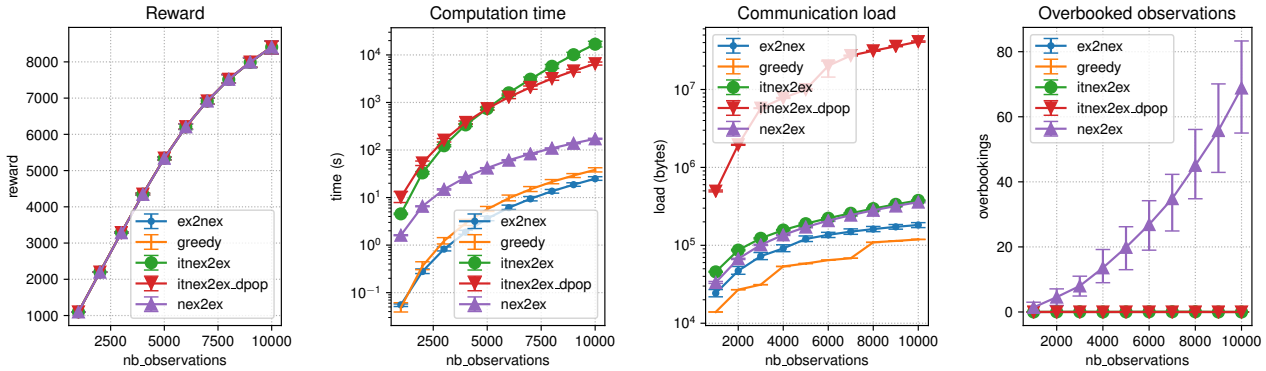

FIGURE 3 – Résultats des algorithmes étudiés sur des problèmes réalistes de grande taille.

EOS. Dans un tel cas, itnex2ex_dpop devient encore plus pertinent.

7 Conclusions

Ce papier étudie pour la première fois l'utilisation de techniques distribuées et multi-agents pour résoudre le nouveau EOSCSP, en gardant à l'esprit la nécessité de limiter la divulgation d'informations entre les utilisateurs. Nous avons défini les composantes essentielles de l'EOSCSP et proposé un codage MILP simple pour résoudre de manière optimale ces problèmes. Ce système est malheureusement inutilisable dans la pratique, même seulement sur de petites instances. Nous avons donc proposé un algorithme glouton et rapide pour résoudre l'EOSCSP. Nous avons conçu plusieurs algorithmes distribués (ex2nex, nex2ex, itnex2ex et itnex2ex_dcop) avec différentes propriétés, notamment concernant la divulgation d'informations. itnex2ex_dcop offre des solutions équivalentes aux meilleurs algorithmes sur des problèmes très conflictuels, tout en préservant la confidentialité des informations d'observation des utilisateurs exclusifs. Ceci est dû au schéma de communication et à l'utilisation de DCOPs pour coordonner les utilisateurs exclusifs lorsqu'ils décident des observations à planifier. Cela a un coût : une charge de communication et un temps de calcul plus élevés pour évaluer la récompense que représente l'intégration d'une observation dans un plan donné. Pourtant, cette technique est entièrement distribuable et peut tirer profit d'une exécution simultanée. Sur des problèmes réalistes à grande échelle, la qualité de la solution est également bonne, bien que ces problèmes nécessitent moins de coordination car la probabilité de chevauchement des observations est plus faible. Les instances DPOP impliquent toujours de nombreuses observations d'utilisateurs exclusifs, ce qui rend le calcul de la fonction d'évaluation π et l'exécution de DPOP encore coûteux.

Ce travail nous permet d'identifier plusieurs perspectives, notamment le développement de solveurs DCOP dédiés et adaptés à la spécificité des EOSCSP, par exemple l'utilisation de la fonction d'évaluation π, qui peut résulter d'un processus d'apprentissage, au lieu d'une évaluation systématique de chaque alternative. Deuxièmement, nous envisageons de considérer

un EOSCSP comme un problème d'optimisation de consensus où les utilisateurs exclusifs et le planificateur central se coordonnent en utilisant des méthodes de décomposition duale pour converger vers un accord sur la planification des observations. Enfin, nous travaillons actuellement à l'intégration des incertitudes sur le succès des observations dans le processus de décision.

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] J. Bonnet, M.P. Gleizes, E. Kaddoum, S. Rainjonneau, and G. Flandin. Multi-satellite mission planning using a self-adaptive multi-agent system. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 11–20, Boston, 2015. IEEE.

[2] Kerri L. Cahoy and Andrew K. Kennedy. Initial Results from ACCESS : An Autonomous CubeSat Constellation Scheduling System for Earth Observation. In *Proc. of the 31st Annual AIAA/USU Conference on Small Satellites*, 2017.

[3] Doo-Hyun Cho, Jun-Hong Kim, Han-Lim Choi, and Jaemyung Ahn. Optimization-based scheduling method for agile earth-observing satellite constellation. *Journal of Aerospace Information Systems*, 15(11) :611–626, 2018.

[4] A. Farinelli, A. Rogers, A. Petcu, and 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)*, pages 639–646, 2008.

[5] F. Fioretto, E. Pontelli, and W. Yeoh. Distributed constraint optimization problems and applications : A survey. *Journal of Artificial Intelligence Research*, 61 :623–698, 2018.

[6] Lei He, Liu Xiaolu, Gilbert Laporte, Ying-Wu Chen, and Yingguo Chen. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers and Operations Research*, 100 :12–25, 07 2018.

[7] M. Johnston. Scheduling nasa's deep space network : Priorities, preferences, and optimization. 2020.

[8] Michel Lemaître, Gérard Verfaillie, Frank Jouhaud, Jean-Michel Lachiver, and Nicolas Bataille. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology*, 6(5) :367 – 381, 2002.

[9] R.T. Maheswaran, J.P. Pearce, and 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)*, pages 432–439, 2004.

[10] P.J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT : Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 2005.

[11] Sreeja Nag, Alan Li, Vinay Ravindra, Marc Sanchez Net, Kar-Ming Cheung, Rod Lammers, and Brian Bledsoe. Autonomous Scheduling of Agile Spacecraft Constellations with Delay Tolerant Networking for Reactive Imaging. In *Proceedings of the International Conference on Automated Planning and Scheduling SPARK Workshop*, 2019.

[12] Peng Feng, Hao Chen, Shuang Peng, Luo Chen, and Longmei Li. A method of distributed multi-satellite mission scheduling based on improved contract net protocol. In *2015 11th International Conference on Natural Computation (ICNC)*, pages 1062–1068, Zhangjiajie, China, 2015. IEEE.

[13] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 266–271, 2005.

[14] Pierre Rust, Gauthier Picard, and Fano Ramparany. pyDCOP, a DCOP library for IoT and dynamic systems. In *International Workshop on Optimisation in Multi-Agent Systems (OptMAS@AAMAS 2019)*, 2019.

[15] Vishwa Shah, Vivek Vittaldev, Leon Stepan, and Cyrus Foster. Scheduling the world's largest earth-observing fleet of medium-resolution imaging satellites. *IWPSS*, 2019.

[16] P. K. Sinha and A. Dutta. Multi-satellite task allocation algorithm for earth observation. In *2016 IEEE Region 10 Conference (TENCON)*, pages 403–408, New York, New York, US, 2016. IEEE.

[17] C. Sun, X. Wang, and X. Liu. Distributed satellite mission planning via learning in games. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4381–4386, New York, New York, US, 2018. IEEE.

[18] Panwadee Tangpattanakul, Nicolas Jozefowiez, and Pierre Lopez. A multi-objective local search heuristic for scheduling Earth observations taken by an agile satellite. *European Journal of Operational Research*, 245(2) :542–554, September 2015.

[19] Alex Elkjær Vasegaard, Mathieu Picard, Florent Hennart, Peter Nielsen, and Subrata Saha. Multi criteria decision making for the multi-satellite image acquisition scheduling problem. *Sensors*, 20(5) :1242, 2020.

[20] J. G. Walker. Satellite Constellations. *Journal of the British Interplanetary Society*, 37 :559, December 1984.

[21] Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. Agile earth observation satellite scheduling over 20 years : formulations, methods and future directions. *CoRR*, abs/2003.06169, 2020.

[22] W. Yang, Y. Chen, R. He, Z. Chang, and Y. Chen. The Bi-objective Active-Scan Agile Earth Observation Satellite Scheduling Problem : Modeling and Solution Approach. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6, July 2018.

[23] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout : Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2) :55–87, January 2005.