Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions

Gauthier Picard

Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions

Gauthier Picard
ONERA/DTIS, Université de Toulouse
gauthier.picard@onera.fr

Abstract

We investigate the use of multi-agent allocation 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 market-based techniques and a distributed problem solving technique based on Distributed Constraint Optimization (DCOP), where agents cooperate to allocate requests without sharing their own schedules. These contributions are experimentally evaluated on randomly generated EOSCSP instances based on real large-scale or highly conflicting observation order books.

1 Introduction

Recent years have shown a large increase in the development of satellite constellations. Instead of considering individual satellites, they take advantage of a group of satellites, some of them often sharing the same orbital planes, to provide richer services like positioning, telecommunication or Earth observation (Walker 1984). With few satellites in a constellation (*e.g.* two in the PLEIADES project (Lemaître et al. 2002)), and in low or medium Earth orbits (altitude inferior to 35,000km), no region on Earth is permanently covered by the constellation at any time. So, the main motivation to increase the size of these constellations is to allow to capture with a high reactivity any point on Earth, as the Planet company is doing with more than 150 Earth Observation Satellites (EOS) (Shah et al. 2019). But, operating numerous EOS requires improving cooperation between the assets and on-board autonomy in order to make the best use of the system, which becomes a highly combinatorial task. Besides their growing number, constellations' composition is evolving too. Recent technological advances allow the production and deployment of agile EOS able to change their orientation, and to provide multiple types of image shooting with multiple sensors. While providing richer services to multiple users, this adds many degrees of freedom and decision variables to schedule EOS activity, and opens many challenges (Wang et al. 2020). Among these challenges, we focus on the collective scheduling of observations on a set of satellites on which some users have *exclusive access to some orbit portions*, using distributed techniques, as to spread decisions among the different users of the constellation. This anwers to strong user expectations to benefit on the one hand from the advantages of a system shared between several stakeholders (reduction of costs compared to a very expensive global system) and on the other hand from the advantages of a proprietary system (ability to do what one wants with the satellite and potentially without disclosing it to others). While the literature about multi-satellite scheduling is rich, as confirmed by a recent review paper (Wang et al. 2020), considering satellite constellations as shared resources requiring multiple users to coordinate as to allocate tasks within exclusive orbit portions is a completely novel problem, we address in this paper, as illustrated in Figure 1.

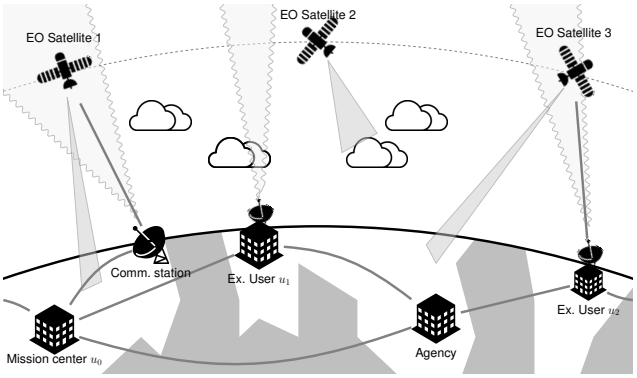

Figure 1: An Earth Observation system composed of a main mission center u_0, distributed stations (with ranges), agencies emitting observation requests (to mission center), EOS (with image footprint), communication satellites (linking EOS), and exclusive users with their own ground stations.

In (Phillips and Parra 2021), marked-based approaches are proposed to allocate observation tasks to a set of satellite, where each satellite is managed by a different mission center. Mission centers coordinate their allocation using auction-based protocols, by bidding on the open observations depending on the impact on the on-board plan and its reward (valued using the incidence angle of the scheduled observations). Contrary to this approach, in this study,

the distribution is related to some exclusive users having full control on some orbit portions (using full direct tasking operation) or having bought some orbit portions outside direct communication, on which they have full priority to schedule observations. Here, the fact that schedules cannot be performed by a single authority, for privacy reason in exclusive windows, is a strong requirement. This is the reason to provide distributed scheduler where agents coordinate without disclosing their plans, while meeting coupling constraints like satellite capacity or inter-observation configuration time, that could not be guaranteed by non-coordinated schemes where users make their plans in parallel. We will investigate here two different distributed resource allocation and coordination schemes: market-based and DCOP-based (distributed constraint optimization).

Section 2 illustrates and defines Earth Observation Satellite Constellation Scheduling Problem (EOSCSP). Section 3 focuses on centralized solution methods Mixed-Integer Linear Program (MILP) and greedy approach to EOSCSP. Section 4 expounds some market-based approaches to solve EOSCSP, using different auction schemes (PSI, SSI and CBBA), while Section 5 proposes another approach to coordination between exclusive users using distributed constraint optimization (DCOP).

We experimentally evaluate these different algorithms using randomly generated instances, in Section 6. Finally, Section 7 concludes the paper with some perspectives.

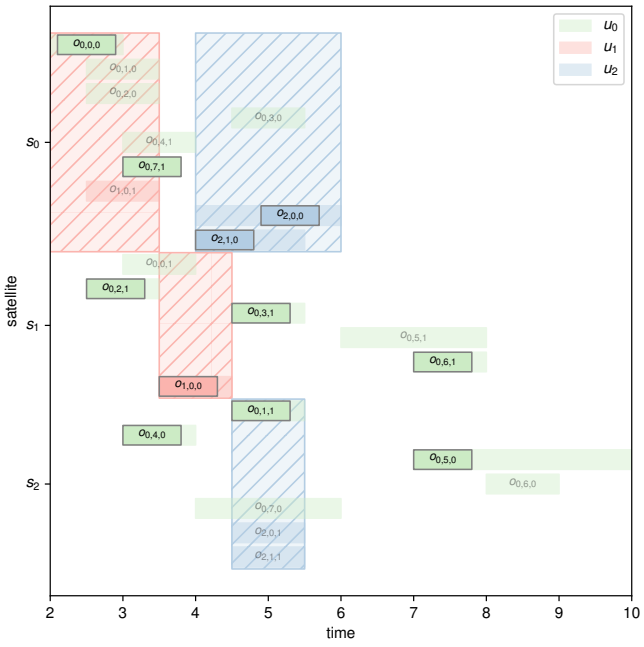

Figure 2: An example with 3 satellites, 2 exclusive users (red and blue) with exclusives (hashed areas), and 1 non-exclusive user (green). Observation time windows appear as transparent surfaces. Solid surfaces represent an optimal solution.

2 EOSCSP Model

This section illustrates the problem we investigate using a sample scenario, and then provides some core definitions.

2.1 Sample Scenario

Figure 2 illustrates a scenario, where we consider: 3 satellites, each having a given planning period (e.g. planning on the next orbit, or on horizons depending on the communication windows between the satellite and ground stations); 1 user u_0 without exclusive orbit portion; 2 users having exclusive orbit portions such that user u_1 owns exclusives on satellite s_0 and on satellite s_1 (hashed red), user u_2 owns exclusives on satellite s_0 and on satellite s_2 (hashed blue); several requests to be performed before a due date, denoted $r_{i,j}$ for the jth request for user i; several observation opportunities (simply observations) per request, denoted $o_{i,j,k}$ for the kth observation for the jth request of the ith user. Only one observation should be planned to fulfill the request on temporal slots depending on the satellites' orbits and the position of zones of interest (slots are represented as transparent areas). More precisely, we consider 2 observations per request, such that observations $o_{1,0,0}$ and $o_{1,0,1}$ are private for user u_1 (in red), observations $o_{2,0,0}$, $o_{2,0,1}$, $o_{2,1,0}$, and $o_{2,1,1}$ are private for user u_2 (in blue), observations $o_{0,j,k}$'s (in green) which are directly requested to the central scheduler u_0 by other clients without exclusives. The proposed solution fulfills all requests, by allowing non exclusive user u_0 to position observation on exclusive orbit portions (e.g. $o_{0,0,0}$ in on u_1's exclusive on satellite s_0). A simplified energy constraint states that a satellite cannot perform more than $n_{\max}$ observations on its scheduling period (here, $n_{\max} = 4$), and there are minimal transition times between two observations o and p, depending on o and p and the date at which the transition is triggered on a given satellite. At the global level, each exclusive user (u_1 or u_2) could have its own scheduling system to manage its exclusive periods, and a central scheduling system (u_0) manages observations $o_{0,j,k}$'s. In the end, every user and the central scheduler have a local scheduling problem to solve. Solving them in a separate manner may lead the central scheduler not to be able to book slots on exclusive orbit portions, while it may improve the solution. Without coordination, and with a non-cooperative management of exclusive slots, the overall schedule might not be optimal, wrt the number of possible scheduled observations. Moreover, exclusive users may gain from this cooperation by making profits from observation scheduled on their orbit portions. Thus, we propose here to coordinate the scheduling processes between users.

2.2 Definitions and Notations

Let's provide the core concepts of this scheduling problem.

Definition 1 *An* Earth Observation Satellite Constellation Scheduling with Exclusives Problem *(or EOSCSP) is defined by a tuple* $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$*, such that* $\mathcal{S}$ *is a set of satellites,* $\mathcal{U}$ *is a set of users,* $\mathcal{R}$ *is a set of requests, and* $\mathcal{O}$ *is a set of observations to schedule to fulfill requests in* $\mathcal{R}$*.*

Definition 2 *A* satellite *is defined as a tuple* $s = \langle t_s^{start}, t_s^{end}, \kappa_s, \tau_s \rangle$ *with* $t_s^{start} \in \mathbb{R}$ *the start time of its orbit plan,* $t_s^{end} \in \mathbb{R}$ *the end time of its orbit plan,* $\kappa_s \in \mathbb{N}^+$ *its ca-*

pacity (i.e. the maximum number of observations during its orbit plan), $\tau_s : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$ the function defining transition times between two given observations.

Definition 3 *A* user *is defined as a tuple $u = \langle e_u, p_u \rangle$ with a (possibly empty) set of exclusive time windows $e_u = \{(s, (t^{start}, t^{end})) \mid s \in \mathcal{S}, [t^{start}, t^{end}] \subseteq [t_s^{start}, t_s^{end}])\} \subset (\mathcal{S} \times (\mathbb{R} \times \mathbb{R}))$, and a priority $p_u \in \mathbb{N}^+$ (the lower the better, used in case of conflict). We note $\mathcal{U}^{\mathsf{ex}}$ (resp. $\mathcal{U}^{\mathsf{nex}}$) the set of users owning (resp. not owning) exclusives.*

We assume here that only one user has no exclusive orbit portion, the central planner, denoted u_0, i.e. $\mathcal{U}^{\mathsf{nex}} = \{u_0\}$, and there is no overlapping exclusive portions.

Definition 4 *A* request *is defined as a tuple $r = \langle t_r^{start}, t_r^{end}, \Delta_r, \rho_r, p_r, u_r, \theta_r \rangle$, with a validity time window defined by $t_r^{start} \in \mathbb{R}$ and $t_r^{end} \in \mathbb{R}$, a duration $\Delta_r \in \mathbb{R}$, a reward $\rho_r \in \mathbb{R}$ if r is fulfilled, a latitude-longitude-altitude position (LLA) to observe p_r, a requester $u_r \in \mathcal{U}$ and a list $\theta_r \in 2^{\mathcal{O}}$ of observation opportunities to fulfill the request.*

θ_r is dynamically computed on current constellation configuration and requested LLA position p_r, since several agile satellites, by changing their orientation may acquire the same position, thus generating several observation opportunities.

Definition 5 *An* observation *is defined as a tuple $o = \langle t_o^{start}, t_o^{end}, \Delta_o, r_o, \rho_o, s_o, u_o, p_o \rangle$, with a validity time window defined by $t_o^{start} \in \mathbb{R}$ and $t_o^{end} \in \mathbb{R}$, a request r_o to which it contributes, a duration $\Delta_o \in \mathbb{R}$ ($\Delta_o = \Delta_{r_o}$), a reward $\rho_o \in \mathbb{R}$ (combined from r_o and information about the weather), a satellite s_o on which this observation can be scheduled, an owner $u_o \in \mathcal{U}$ ($u_o = u_{r_o}$), and a priority $p_o \in \mathbb{N}^+$ ($p_o = p_{r_o}$).*

The difference between request reward and observation reward comes from the fact that, in practice, weather conditions or incidence angle of an observation may increase or decrease the basic reward for a given request. So, our model can consider different rewards, but in this study we only focus on cases where observation rewards are directly inherited from the requests.

Definition 6 *A* solution *to an EOSCSP is a mapping $\mathcal{M} = \{(o, t) \mid o \in \mathcal{O}, t \in [t_o^{start}, t_o^{end}]\}$ assigning a start time to at most one observation per request such that exclusive users have their observations scheduled on their respective exclusive windows, and the overall reward is maximized (sum of the rewards of the scheduled observations):* $\arg\max_{\mathcal{M}} \sum_{(o,t)\in\mathcal{M}} r_o$.

Definition 7 *An* EOSCSP for user u, *denoted $P[u] = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}[u], \mathcal{O}[u] \rangle$ (or EOSCSP$[u]$), is an EOSCSP, sub-problem of another EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$ restricted to requests and observations from u, where $\mathcal{R}[u] = \{r \mid r \in \mathcal{R}, u_r = u\} \subseteq \mathcal{R}$ and $\mathcal{O}[u] = \{o \mid o \in \mathcal{O}, u_o = u\} \subseteq \mathcal{O}$.*

More generally, we note $P[x]$ (resp. $P[x]$) the problem P limited to the only components related to x, x being a request, an observation or a satellite. Later on, we will also use the notations $P[\emptyset|\mathcal{M}]$ (resp. $P[u_l, \ldots, u_m|\mathcal{M}]$) to define the problem (resp. sub-problem for users $u_l, \ldots, u_m$) given some predefined allocation $\mathcal{M}$ of some observations. Moreover, we will use notation $\overline{P}$ to appoint the EOSCSP P, where only requests and related observations that can be scheduled outside exclusive are considered (i.e. observations whose time windows intersect non exclusive orbit portions). Finally, we note the union of two problems $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$ and $P' = \langle \mathcal{S}', \mathcal{U}', \mathcal{R}', \mathcal{O}' \rangle$, $P \cup P' = \langle \mathcal{S} \cup \mathcal{S}', \mathcal{U} \cup \mathcal{U}', \mathcal{R} \cup \mathcal{R}', \mathcal{O} \cup \mathcal{O}' \rangle$.

3 Centralized Problem Solving for EOSCSP

We present here centralized approaches to EOSCSP. First, this planning problem is modeled as a MILP. Decision variables are the following. $x_{s,o} \in \{0, 1\}$ is the decision to perform observation o from satellite s, $t_{s,o} \in \mathbb{R}$ is the start date for the observation o on satellite s, $\beta_{s,o,p} \in \{0, 1\}$ is the precedence between observations on the same satellite, equals to 1 if o is before p on s.

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

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

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

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

$$\beta_{s,o,p} + \beta_{s,p,o} \leq 1 \tag{4}$$

$$t_{s,p} - t_{s,o} \geq \tau_s(o, p) + \Delta_o - \Delta^{\max}_{s,o,p}\beta_{s,o,p},\ \Delta^{\max}_{s,o,p} > 0 \tag{5}$$

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

$$\sum_{o\in\mathcal{O}} x_{s,o} \leq \kappa_s \tag{7}$$

$$\sum_{o\in\theta(r)} x_{s,o} \leq 1 \tag{8}$$

$$x_{s,o} \in \{0, 1\} \tag{9}$$

$$t_{s,o} \in [t_o^{\text{start}}, t_o^{\text{end}}] \subset \mathbb{R} \tag{10}$$

$$\beta_{s,o,p} \in \{0, 1\} \tag{11}$$

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

(2) to (6) ensure precedence of observations and their distance is at least the transition time required on their satellite. (7) enforces the number of observations booked on a satellite does not exceed its capacity. (8) checks at most one observation per request is scheduled. (9) to (11) are domain definitions. This MILP can be solved using off-the-shelf solvers like CPLEX or Gurobi, but they will hardly scale up when dealing with larger problems (e.g. more than 100 observations with 3 satellites and 3 users). To ensure observations from exclusive users have priority over non-exclusive users' observations, their reward must be set to a high value. Thus, the solver will prefer scheduling exclusive observations within their time window instead of scheduling another observation with less priority. While the solution to this problem is optimal, it requires each exclusive user to *fully disclose request information* to the central planner.

As to solve large problems, one approach is to apply a greedy allocation consisting in planning first exclusive users's observations and then more urgent observations, as described in Algorithm 1. In practice, this is the technique

used by most satellite/constellation operators and is a candidate competitor for benchmarking solution methods (Cho et al. 2018; Wang et al. 2020). For doing so, observations are sorted in increasing order on priority and start time criteria (line 2). Then, for each observation in this sorted list, the first free slot on its satellite orbit plan is found (line 4-8). This algorithm is not optimal, but provides very fast solutions. However, as for MILP, this solution requires sharing all the constraints and information with a central planner.

Algorithm 1: Greedy EOSCSP solver

Data: An EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Result: An assignment $\mathcal{M}$

1 $\mathcal{M} \leftarrow \{\}$
2 $\mathcal{O}^{\text{sorted}} \leftarrow \texttt{sort}(\mathcal{O})$
3 $R \leftarrow \{(s, [])\} \mid s \in \mathcal{S}\}$
4 **for** $o \in \mathcal{O}^{sorted}$ **do**
5 $t \leftarrow \texttt{first\_slot}(o, P, R)$
6 **if** $t \neq \emptyset$ **then**
7 $\mathcal{M} \leftarrow \mathcal{M} \cup \{(o, t)\}$
8 $\mathcal{O}^{\text{sorted}} \leftarrow \mathcal{O}^{\text{sorted}} \setminus \theta(r_o)$
9 **return** $\mathcal{M}$

Function `first_slot` *(o,* $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$*, R)*

10 **for** $(s, [t^{start}, t^{end}]) \in \texttt{domains}(o)$ **do**
11 **if** $|R[s]| < \kappa_s$ **then**
12 **if** $R[s] = []$ **then**
13 **if** $t^{end} \geq t^{start} + \Delta_o$ **then**
14 $R[s] = \{(o, (s, t^{\text{start}}))\}$
15 **return** (s, t^{start})
16 **else**
17 $i \leftarrow 0$
18 **while** $i \leq |R[s]|$ **do**
19 $t^{\text{start}\prime} \leftarrow t^{\text{start}}$
20 **if** $i > 0$ **then**
21 $(o_{i-1}, (s, t_{i-1})) \leftarrow R[s][i-1]$
22 $t^{\text{start}\prime} \leftarrow \max(t^{\text{start}}, t_{i-1} + \Delta_{o_{i-1}} + \tau_{s,o_{i-1},o})$
23 **if** $t^{start\prime} + \Delta_o \leq t^{end}$ **then**
24 **if** $i = |R[s]|$ **then**
25 $t^{\text{upper}} \leftarrow t^{\text{end}}$
26 $t^{\text{end}\prime} \leftarrow t^{\text{start}\prime} + \Delta_o$
27 **else**
28 $(o_i, (s, t_i)) \leftarrow R[s][i]$
29 $t^{\text{upper}} \leftarrow t_i$
30 $t^{\text{end}\prime} \leftarrow t^{\text{start}\prime} + \Delta_o + \tau_{s,o,o_i}$
31 **if** $t^{start\prime} < t^{end\prime} \leq t^{upper}$ **then**
32 $R[s] = \texttt{insert}(R[s], (o, (s, t^{\text{start}\prime})), i)$
33 **return** $(s, t^{start\prime})$
34 $i{+}{+}$
35 **return** $\emptyset$

4 Auction-based Coordination for EOSCSP

One vision to allocate resources and/or tasks between several agents (here, our exclusive users) consists in market-based approaches, that have proven their flexibility, efficiency, fairness, and privacy-preservation of users' plans and resources. In multi-robot task allocation problems, such approaches are used to allocate tasks to robots, and integrate them into their plans (Dias et al. 2006). In our setting, one could consider allocating requests to satellites by such market-based mechanisms, as proposed in (Phillips and Parra 2021), with the difference that distribution is not related to satellites, but to exclusive users and their exclusive orbit portions. This is the approach we follow in this section. But first, let's introduce the auction-based mechanisms we will implement.

4.1 Some Background on Market-based Allocation

A generic task allocation framework consists in a set of resources and a set of tasks to be performed by resources. The objective is to assign tasks to resources so that it maximizes some objective (e.g. the number of assigned tasks, or the sum of the rewards of the tasks). So this is classical allocation problem that can be modeled as a MILP, as seen in previous section. Now, the idea is that the requests to be scheduled are open for bidding by an *auctioneer*. *Bidders* (the exclusive users) valuate the requests depending on their current plan, and bid for some requests, as illustrated in Figure 3. The most expensive computations in this process are the bidding step by each bidder, which can have an exponential number of bundles to valuate, and the winner determination problem (WDP) which amounts to solving an Integer Linear Program with a potentially exponential size, and falls into the combinatorial auction (CA) frameworks (Cramton, Shoham, and Steinberg 2010).

According to literature on multi-robot task allocation (Dias et al. 2006) and multi-satellite observation allocation (Phillips and Parra 2021), to overcome these computational limits, the classical relaxation consists in only allowing bidding on item (and not on bundles). When bidders bid on the whole set of items in parallel, we fall into PSI framework (Koenig et al. 2006). When the auctioneer announces items iteratively, and bidders build their bid knowing the previous item allocation, we fall into the SSI framework (Koenig et al. 2006). In general PSI has very good performances with very limited computation time, while PSI solution quality are often limited, since bidders cannot easily reason on bundles. More recently, consensus-based bundle algorithm (CBBA) combines ideas from auctions and consensus to converge faster than SSI while yielding similar solutions and having the benefits of traditional consensus algorithms (Choi, Brunet, and How 2009). CBBA is a fully distributed solution to implement a computationally cheap variant of combinatorial auctions (CA). Each bidder constructs a unique bundle of items it wishes to be assigned to, with respect to the marginal cost associated with the inclusion of the considered item into its current bundle. Then during the consensus phase, the bidders compare their bids with their teammates

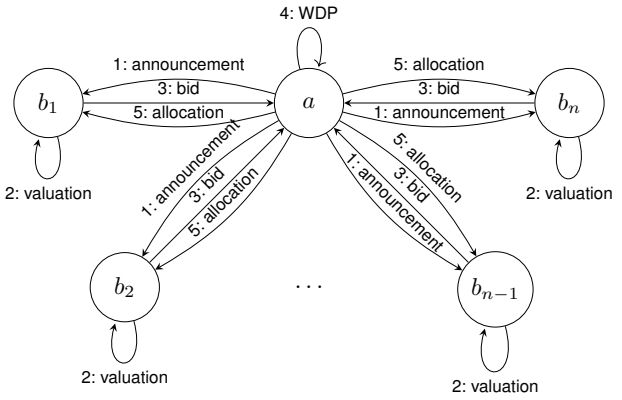

Figure 3: A sample auction process with one auctioneer a and n bidders b_i, following five main steps: (1) announcement of the items to allocate, (2) valuation of the items or bundles by each bidder, (3) communication of the computed bids, (4) winner determination problem solving, and (5) allocation of items to bidders.

bids. If a robot is outbid on an item t, it drops the item and all the items added after it, as the exclusion of t made the valuation of their marginal cost obsolete. This algorithm have been extensively studied and modified to improve its performances and adapt it to specific scenarios, like multi-satellite observation allocation (Phillips and Parra 2021).

4.2 Mapping EOSCSP to Auctions

Mapping an EOSCSP P to a market-based allocation problem is quite straightforward. Bidders are exclusives users in $\mathcal{U}^{\text{ex}}$, and items are non exclusive requests in $\mathcal{R}$ emitted by the central planner u_0, playing the role of auctioneer. The idea is that each exclusive user u computes an initial plan $\mathcal{M}_u$ with its own requests by solving $P[u]$. Then, u_0 announces the requests, either as a whole (for PSI and CBBA) or iteratively (for SSI). Each exclusive user u valuates each single request with function `bid` (for PSI and SSI) or bundle with function `bundle` (for CBBA) by computing the marginal cost to integrate the given item or bundle x in its current plan. We redirect the reader to the original CBBA paper for more details about the bundle construction (Choi, Brunet, and How 2009). $\texttt{bid}(r, \mathcal{M}_u)$ simply amounts to solve $P[u] \cup P[r]$ and to assess the difference with the current plan $\mathcal{M}_u$. It returns the bid itself $\mathcal{B}_u[r]$ (best marginal cost) and the schedule for one observation to fulfill r, $\sigma_u[r] = (o, t)$. The bids (on single items or bundles) are then sent to the auctioneer u_0 (for PSI and SSI) which determines the winners, or to the other bidders sharing interest on the same request, namely $\mathcal{N}_u$, to find a consensus (for CBBA). Once the winners are determined, requests are allocated to the winners. If there remain some non allocated requests, u_0 attempts to schedule them outside any exclusive window. These processes are sketched in Algorithms 2, 3 and 4.

In PSI and SSI, the $\oplus$ operator is used to add $\sigma_u[r] = (o, t)$ in the current plan. Depending on the setting, it can be

Algorithm 2: psi EOSCSP solver

Data: An EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Result: An assignment $\mathcal{M}$

$\mathcal{M}_{u_0} \leftarrow \emptyset$
for each $u \in \mathcal{U}^{ex}$ **do concurrently**
  $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
  for each $r \in \mathcal{R}$ **do** $\mathcal{B}_u[r], \sigma_u[r] \leftarrow \texttt{bid}(r, \mathcal{M}_u)$
  `// send` $\mathcal{B}_u, \sigma_u$ `to` u_0
for each $r \in \mathcal{R}$ **do**
  $w \leftarrow \arg\max_{u \in \mathcal{U}^{\text{ex}}} \{\mathcal{B}_u[r]\}$
  $\mathcal{M}_{u_0} \leftarrow \mathcal{M}_{u_0} \cup \{\sigma_w[r]\}$
  $\mathcal{M}_w \leftarrow \mathcal{M}_w \oplus \sigma_w[r]$ `// send` $\mathcal{M}_w[r]$ `to` w
$\mathcal{M}_{u_0} \leftarrow \texttt{solve}(\overline{P[u_0|\mathcal{M}_{u_0}]})$
return $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

Algorithm 3: ssi EOSCSP solver

Data: An EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Result: An assignment $\mathcal{M}$

$\mathcal{M}_{u_0} \leftarrow \emptyset$
for each $u \in \mathcal{U}^{ex}$ **do concurrently** $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
for each $r \in \texttt{sorted}(\mathcal{R})$ **do**
  for each $u \in \mathcal{U}^{ex}$ **do**
    $\mathcal{B}_u[r], \sigma_u[r] \leftarrow \texttt{bid}(r, \mathcal{M}_u)$
    `// send` $\mathcal{B}_u[r], \sigma_u[r]$ `to` u_0
  $w \leftarrow \arg\max_{u \in \mathcal{U}^{\text{ex}}} \{\mathcal{B}_u[r]\}$
  $\mathcal{M}_{u_0} \leftarrow \mathcal{M}_{u_0} \cup \{\sigma_w[r]\}$
  $\mathcal{M}_w \leftarrow \mathcal{M}_w \oplus \sigma_w[r]$ `// send` $\mathcal{M}_w[r]$ `to` w
$\mathcal{M}_{u_0} \leftarrow \texttt{solve}(\overline{P[u_0|\mathcal{M}_{u_0}]})$
return $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

Algorithm 4: cbba EOSCSP solver

Data: An EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Result: An assignment $\mathcal{M}$

$\mathcal{M}_{u_0} \leftarrow \emptyset$
for each $u \in \mathcal{U}^{ex}$ **do concurrently** $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
for each $r \in \texttt{sorted}(\mathcal{R})$ **do**
  for each $u \in \mathcal{U}^{ex}$ **do**
    $\mathcal{N}_u \leftarrow \texttt{candidates}(r)$
    $\mathcal{R}_u \leftarrow \mathcal{R}_u \cup \{r\}$
while *conflict* **do**
  for each $u \in \mathcal{U}^{ex}$ **do concurrently**
    $\mathcal{B}_u, \mathcal{W}_u, \mathcal{T}_u \leftarrow \texttt{bundle}(u)$
    `// send` $\mathcal{B}_u, \mathcal{W}_u, \mathcal{T}_u$ `to` $\mathcal{N}_u$
  for each $u \in \mathcal{U}^{ex}$ **do concurrently**
    `// solve conflicts and determine` $\mathcal{M}_u$ `(see (Choi, Brunet, and How 2009))`
for each $u \in \mathcal{U}^{ex}$ **do**
  $\mathcal{M}_{u_0} \leftarrow \mathcal{M}_{u_0} \cup \{(o, t) | (o, t) \in \mathcal{M}_u, u_o = u_0\}$
$\mathcal{M}_{u_0} \leftarrow \texttt{solve}(\overline{P[u_0|\mathcal{M}]})$
return $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

a simple aggregation if there is no conflict, or may require removing some already planned observations with lower reward. In SSI and CBBA, requests are sorted before looping over. This sorting can be done wrt due date, reward, or any combination of criteria. In the experiments, we will use the due date.

5 DCOP-based Coordination for EOSCSP

Another approach to implement the allocation of requests between the multiple candidate exclusive users is to adopt a distributed constraint optimization vision. We devise here a cooperation mechanism between exclusive users to coordinate their scheduling process, by exchanging messages to reach an agreement on request allocations while meeting the coupling constraints, such as the capacity constraints.

5.1 Some Background on DCOP

One way to model inter-agent coordination problems is to formalize them as distributed constraint optimization problems (DCOP) (Petcu and Faltings 2005).

Definition 8 *A discrete* Distributed Constraint Optimization Problem *(or DCOP) is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$, where: $\mathcal{A} = \{a_1, \ldots, a_{|A|}\}$ is a set of agents; $\mathcal{X} = \{x_1, \ldots, x_n\}$ are variables owned by the agents; $\mathcal{D} = \{\mathcal{D}_{x_1}, \ldots, \mathcal{D}_{x_n}\}$ is a set of finite domains, such that variable x_i takes values in $\mathcal{D}_{x_i} = \{v_1, \ldots, v_k\}$; $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of soft constraints, where each c_i defines a cost $\in \mathbb{R}^+ \cup \{+\infty\}$ for each combination of assignments to a subset of variables (a constraint is initially known only to the agents involved); $\mu : \mathcal{X} \rightarrow \mathcal{A}$ is a function mapping variables to their associated agent; $f : \prod \mathcal{D}_{x_i} \rightarrow \mathbb{R}$ is an objective function, representing the global cost of a complete variable assignment. The optimization objective is represented by function f, which, in general, is considered as the sum of costs: $f = \sum_i c_i$. A* solution *to a DCOP P is a complete assignment to all variables. A solution is* optimal *if it minimizes f.*

DCOP have been widely studied and applied in many areas of reference (Fioretto, Pontelli, and Yeoh 2018). They have many interesting properties: (i) focus on decentralized approaches where agents negotiate a joint solution through local message exchanges; (ii) exploitation of the domain structure (by encoding it in constraints) to address hard computational problems; (iii) wide variety of solution methods ranging from exact methods to heuristic and approximate techniques; such as, for example, ADOPT (Modi et al. 2005), DPOP (Petcu and Faltings 2005), MaxSum (Farinelli et al. 2008), DSA (Zhang et al. 2005) or MGM (Maheswaran, Pearce, and Tambe 2004), to name only the most famous.

5.2 Coordinating Exclusive Users with DCOP

As for combinatorial auctions, using DCOPs for allocating all requests as a whole is too computationally expensive to be used. So we will use the same idea than SSI, and consider requests sequentially, and coordinate exclusive user for each request using a DCOP solver, instead of auctions in SSI. This lets the non exclusive users coordinate to choose which one will fulfill it by scheduling an observation in its exclusive time windows. Algorithm 5 sketches this method, coined s_dcop (for sequential DCOP). First, exclusive users also solve their own local sub-problem concurrently (line 1). Then, for each request r in the ordered list of remaining requests (line 2), a new DCOP instance is collectively built between the exclusive users (line 3), and then solved (line 4) using any DCOP solver available (DPOP in our experiments). Once all requests have been considered, u_0 gathers the sub-solutions to build its own final solution, by scheduling as many observation outside exclusive time windows as possible (line 6-7). Note that, the inner plan of each exclusive user remains private, and only non exclusive schedule observation are communicated to u_0 (plus some extra information to handle inter-observation transition times).

Algorithm 5: s_dcop EOSCSP solver

Data: An EOSCSP $P = \langle \mathcal{S}, \mathcal{U}, \mathcal{R}, \mathcal{O} \rangle$
Result: An assignment $\mathcal{M}$
1 **for each** $u \in \mathcal{U}^{\text{ex}}$ **do concurrently** $\mathcal{M}_u \leftarrow \texttt{solve}(P[u])$
2 **for each** $r \in \texttt{sort}(\mathcal{R})$ **do**
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 $\quad$ **for each** $u \in \mathcal{U}^{\text{ex}}$ **do concurrently**
6 $\quad\quad \mathcal{M}'_u \leftarrow \{(o,t) \in \mathcal{M}_u | u_o \in \mathcal{U}^{\text{nex}}\}$
$\quad\quad$ // send $\mathcal{M}'_u$ to u_0
7 $\mathcal{M}_{u_0} \leftarrow \texttt{solve}(\overline{P[u_0| \bigcup_{u \in \mathcal{U}^{\text{ex}}} \mathcal{M}'_u]})$
8 **return** $\bigcup_{u \in \mathcal{U}} \mathcal{M}_u$

5.3 DCOP Model

Let's specify now the DCOP instance to be built in line 3 of Algorithm 5 for a given request r, and a current scheduling $(\mathcal{M}, \mathcal{M}_{u_1}, \ldots, \mathcal{M}_{u_n})$, as required in Definition 8. Straightforwardly, the set of agents is the set of exclusive users which can potentially schedule the current request r:

$$\mathcal{A} = \{u \in \mathcal{U}^{\text{ex}} | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \exists o \in \theta_r \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \emptyset\} \quad (12)$$

We note $\mathcal{O}[u]^r = \{o \in \theta r | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \emptyset\}$ these observations related to request r that can be scheduled on u's exclusives. Each such agent will own binary decision variables, one for each observation $o \in \mathcal{O}[u]^r$ and exclusive e in its exclusives e_u, stating whether it schedules o in e or not:

$$\mathcal{X} = \{x_{e,o} | e \in \textstyle\bigcup_{u \in \mathcal{A}} e_u, o \in \mathcal{O}[u]^r\} \quad (13)$$

$$\mathcal{D} = \{\mathcal{D}_{x_{e,o}} = \{0,1\} | x_{e,o} \in \mathcal{X}\} \quad (14)$$

The mapping μ associates each variable $x_{e,o}$ to e's owner.

Constraints should check that at most one observation is scheduled per request (15), that satellites are not overloaded (16), that at most one agent serves the same observation (17).

$$\textstyle\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \quad \forall u \in \mathcal{X}, \forall o \in \mathcal{O}[u]^r \quad (15)$$

$$\textstyle\sum_{o \in \{o \in \mathcal{O}[u]^r | u \in \mathcal{A}, s_o = s\}, e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq \kappa_s^*, \ \forall s \in \mathcal{S} \quad (16)$$

with κ_s^* being the current capacity of s given the already scheduled observations in $\mathcal{M}, \mathcal{M}_{u_1}, \ldots, \mathcal{M}_{u_n}$.

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \quad \forall o \in \mathcal{O} \tag{17}$$

Beside, the cost to integrate an observation in the current user's schedule should be assessed to guide the optimization process. We thus add soft constraint to each $x_{e,o}$:

$$c(x_{e,o}) = \pi(o, \mathcal{M}_{u_o}), \quad \forall x_{e,o} \in \mathcal{X} \tag{18}$$

where π evaluates the best cost obtained when scheduling o and any combination of observations from $\mathcal{M}_{u_o}$, as to consider all possible revisions of u_o's current schedule. Practically, instead of computing π each time, some constraint compilation can be used to assess all these combinations only once. These exponential number of alternatives are evaluated using polynomial greedy algorithm. To sum up:

$$\mathcal{C} = \{(15), (16), (17), (18)\} \tag{19}$$

6 Experimental Evaluation

Experiments aim to analyze the performances of the investigated algorithms with a growing number of requests (and observations). They are coded in Python 3.7 and executed on 20-core Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz, 62GB RAM, Ubuntu 18.04.5 LTS. We ran 30 instances of randomly generated EOSCSP with seed in [0:29] for each problem size, and plot the average, with [0.05, 0.95] confidence. The `solve` procedure used in psi, ssi, cbba and s_dcop is the greedy algorithm. The DCOP algorithm used by s_dcop is the DPOP implementation from pyDCOP (Rust, Picard, and Ramparany 2019). Randomly generated values are uniformly chosen within provided intervals. The computation time reported latter is a centralized computation time (no real distribution over several computers).

Highly conflicting problems. We evaluate the algorithms on very conflicting small-scale problems (5 min planning horizon). We generate EOSCSPs with 3 satellites with a capacity of 20 observations, 4 exclusive users emitting 2 to 20 requests each, 8 exclusive portions per user with a random duration in [15:20], a central planner emitting 8 to 80 requests, 10 observation opportunities per request of duration 5 that can be scheduled in a time window with duration in [10:20], and a reward in [10:50:10] for exclusive user, and in [1:5] for central planner. Satellites' time window is [0, 300]. Transition times between observations are uniformly equals to 1. Exclusives are randomly positioned, while ensuring they do not overlap. Observation time windows are randomly positioned, as to ensure they are either included in one exclusive, or outside any exclusive. There are many observation overlaps, and as many requests from central planner than all requests from the exclusive users.

Figure 4 shows the results for this setting. Reward-wise, all the distributed algorithms except itnex2ex are almost as good as greedy, which is our baseline. Still, cbba and s_dcop provides the best distributed solutions. The decline with growing number of observations is due to the satellites' capacity saturation. s_dcop and cbba's performances are at the cost of extra computation time, while remaining reasonable (approx. 1000 seconds), contrary to optimal solver (e.g. CPLEX) that cannot solve instances with more than 100 observations (not displayed here). s_dcop's higher computation time results from pre-computing function π and the underlying DPOP solving procedure. cbba computational overhead is due to bundle valuation. At some point (problems larger than 750 observations to schedule), cbba requires more time to compute than s_dcop. This is due to the exponentially growing number of bundles to consider and the fact that at this size, with such a conflicting setting, the cbba neighborhood network is a complete graph, meaning that each user has to resolve conflicts with all the other users. Communication-wise, psi exchanges few large messages, since all requests are communicated to all users, resulting in exchanging more 10Mb in larger instances. ssi and s_dcop exchange numerous messages of smaller size (only sending bids on requests of interest), due to the sequential process they follow. On its side, cbba exchanges fewer messages of small size (approx. total 30kB in large instances), which makes it a very relevant candidate in distributed settings, with good compromise between solution quality and communication load. If reactiveness is a requirement, ssi remains the best candidate.

Realistic problems. Here, we generate large-scale EOSCSPs, with realistic parameters, with respect with order books provided by our partners, to schedule thousands of observations in a 6-hour planning horizon. We generate instances as previously but with 8 satellites with a capacity of 500 observations, 5 exclusive users with 20 to 100 requests each, 10 exclusive orbit portions per user with a duration in [300:600], 1 central planner with 25 to 250 requests, 5 observation opportunities per request of duration of 20 that can be scheduled in a time window with duration in [40:60] included in an exclusive windows (there is no request outside exclusive windows in this setting), and the planning time window is [0, 21600].

Figure 5 shows results for this setting. All algorithms provide good quality solutions equivalent to greedy. The results obtained in this setting, only focusing on observations inside exclusive windows, confirm the performances of the different benchmarked algorithm, except that here cbba does require more time to compute than s_dcop. This is due to the fact that these larger instances are less conflicting, and that the neighborhoods are no longer complete. Let's note that both s_dcop and cbba are very distributed in nature, and performs many computation concurrently. Therefore there is room for computation speedup in real distributed settings.

7 Conclusion and Synthesis

This paper investigated for the first time the use of distributed and multi-agent techniques to solve the novel EOSCSP, keeping in mind the need to limit information disclosure between users. We defined core components of

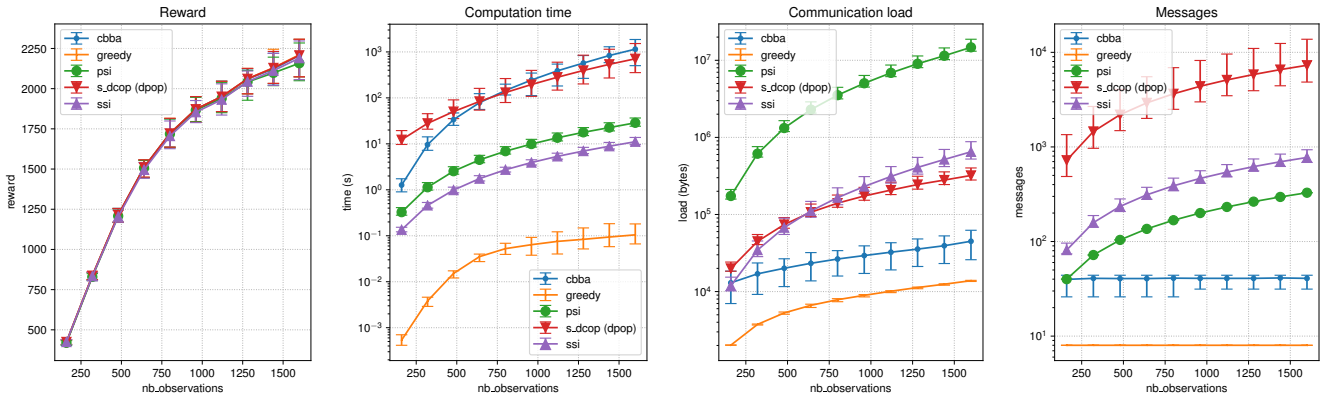

Figure 4: Results for the investigated distributed solution methods on highly conflicting small-scale problems.

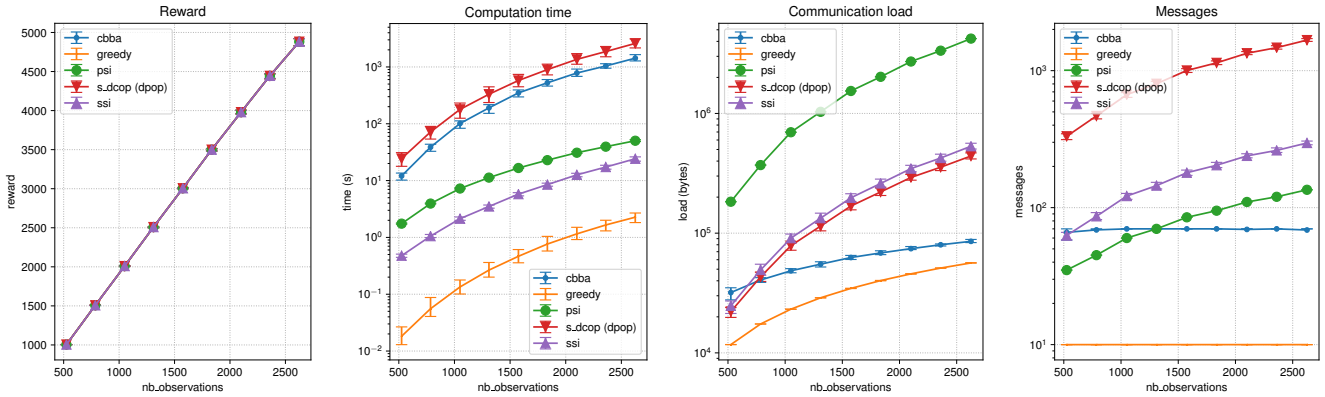

Figure 5: Results for the investigated distributed solution methods on problems with realistic large-scale order books.

EOSCSP, and proposed a straightforward MILP encoding to optimally solve such problems. This is unfortunately non usable in practice, even on small instances. We thus proposed a greedy and fast algorithm to solve EOSCSP. We devised and implemented several distributed algorithms (psi, ssi, cbba and s_dcop), all keeping the inner user plans private. s_dcop and cbba provides solutions equivalent to the best evaluated algorithms on over-conflicting problems. This has a cost: higher communication load and computation time to assess the reward to integrate an observation in a given schedule. Yet, these techniques are fully distributable, and may gain from concurrent execution. On realistic large scale problems, the solution quality is still very good wrt. greedy. While, these problems still require less coordination because the probability for overlapping observations is smaller, EOSCSP still implies numerous observations from exclusive users, which makes the computation of the s_dcop evaluation function π and the construction of the cbba bundles expensive. A good compromise is thus to use ssi in larger settings, since computation time, communication-load are very limited, while providing good quality solutions. Note that this investigation was also a very good terrain for confronting DCOP-based and Market-based techniques, which are most often not compared in the literature.

This work raises several perspectives, notably the development of dedicated DCOP or CBBA solvers adapted to EOSCSP specificity, e.g. the use of the evaluation function π or the construction of bundles, that may result from a learning process, instead of a systematic assessment of every alternative. On may also consider devising dedicated bidding language to assess bundles and perform the winner determination problem in a efficient manner. Finally, we are currently working on integrating uncertainties about observation success into the decision process, which leads to even more complex problems to solve.

Acknowledgments

This work has been performed with the support of the French government in the context of the "Programme d'Invertissements d'Avenir", namely by the BPI PSPC project "LiChIE", coordinated by Airbus Defence and Space.

References

Cho, D.-H.; Kim, J.-H.; Choi, H.-L.; and Ahn, J. 2018. Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation. *Journal of Aerospace Information Systems* 15(11): 611–626. doi:10.2514/1.I010620. URL https://doi.org/10.2514/1.I010620.

Choi, H.; Brunet, L.; and How, J. P. 2009. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Trans. Robotics* 25(4): 912–926. doi:10.1109/TRO.2009.2022423.

Cramton, P.; Shoham, Y.; and Steinberg, R., eds. 2010. *"Combinatorial Auctions"*. MIT Press.

Dias, M.; Zlot, R.; Kalra, N.; and Stentz, A. 2006. Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE* 94(7): 1257–1270. doi:10.1109/JPROC.2006.876939.

Farinelli, A.; Rogers, A.; Petcu, A.; and Jennings, N. R. 2008. Decentralised Coordination of Low-power Embedded Devices Using the Max-sum Algorithm. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, 639–646. ISBN 978-0-9817381-1-6.

Fioretto, F.; Pontelli, E.; and Yeoh, W. 2018. Distributed Constraint Optimization Problems and Applications: A Survey. *Journal of Artificial Intelligence Research* 61: 623–698.

Koenig, S.; Tovey, C. A.; Lagoudakis, M. G.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A. J.; Meyerson, A.; and Jain, S. 2006. 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*, 1625–1629. AAAI Press. URL http://www.aaai.org/Library/AAAI/2006/aaai06-266.php.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and scheduling observations of agile satellites. *Aerospace Science and Technology* 6(5): 367 – 381. ISSN 1270-9638. doi:https://doi.org/10.1016/S1270-9638(02)01173-2. URL http://www.sciencedirect.com/science/article/pii/S1270963802011732.

Maheswaran, R.; Pearce, J.; and Tambe, M. 2004. Distributed Algorithms for DCOP: A Graphical-Game-Based Approach. In *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems (PDCS)*, 432–439.

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

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

Phillips, S.; and Parra, F. 2021. *A Case Study on Auction-Based Task Allocation Algorithms in Multi-Satellite Systems*. doi:10.2514/6.2021-0185. URL https://arc.aiaa.org/doi/abs/10.2514/6.2021-0185.

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

Shah, V.; Vittaldev, V.; Stepan, L.; and Foster, C. 2019. Scheduling the World's Largest Earth-Observing Fleet of Medium-Resolution Imaging Satellites. *IWPSS* .

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

Wang, X.; Wu, G.; Xing, L.; and Pedrycz, W. 2020. Agile Earth observation satellite scheduling over 20 years: formulations, methods and future directions. *CoRR* abs/2003.06169. URL https://arxiv.org/abs/2003.06169.

Zhang, W.; Wang, G.; Xing, Z.; and Wittenburg, L. 2005. Distributed Stochastic Search and Distributed Breakout: Properties, Comparison and Applications to Constraint Optimization Problems in Sensor Networks. *Artificial Intelligence* 161(1-2): 55–87. doi:10.1016/j.artint.2004.10.004. URL http://dx.doi.org/10.1016/j.artint.2004.10.004.