Multi-Agent Consensus-based Bundle Allocation for Multi-Mode Composite Tasks

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

ABSTRACT

We consider agents having to schedule tasks within time slots they own on some resources. We focus on composite tasks that require multiple atomic tasks to be completed, potentially slotted into different private plans, and on disjunctive resources that can only perform one task at a time. There are multiple ways (or modes) to fulfill such composite tasks, with more or less atomic tasks to perform. Some non-owner agents may want to schedule such multi-mode composite tasks requiring access to one or more private slots. Slot owners thus have to coordinate as to collectively fulfill the requests, without disclosing their own plans. This scenario is motivated by innovative concepts of operations in Earth observation satellite constellations, where some users own some orbit slots, and plan any observation task they want by directly communicating with the satellites flying within their slots. We address this problem from a multi-agent perspective where agents are slot owners, which make use of a coordination mechanism built upon an auction-based task allocation technique, MM-CBGA, adapting the consensus framework to the case of multi-mode composite requests. The contribution is evaluated and compared to centralized greedy allocations and sequential auctions, using simulated constellations and realistic order books for observations over Europe.

KEYWORDS

Task allocation, Composite tasks, Multiple modes, Consensus-Based Group Allocation, Earth observation satellites

ACM Reference Format:
Gauthier Picard. 2023. Multi-Agent Consensus-based Bundle Allocation for Multi-Mode Composite Tasks. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

The deployment of larger constellations for Earth observation tasks requires solving more and more complex planning and scheduling problems. Indeed, while a larger number of satellites linearly increases the number of opportunities to satisfy complex user requests (e.g. monoscopic, stereoscopic, systematic, periodic) at a higher frequency, it also exponentially increases the size of the resulting scheduling and task allocation problems [22]. Even more constraining, solving such problems should be performed several times a day, in less than few minutes, as to adapt the plan to meteorological conditions, or last-minute high-priority requests, and so that plans are directly sent to the satellites as soon as they are within communication range. And to make it even more complex, some constellations now consider concepts of operations where some users own some orbit slots for a long period of time [16, 17, 20]. Owners fully manage the plans within their orbit slots and keep them private. Any non-owner user wishing to perform some observations requiring access to some private orbit slots needs to make a request to owners, which have to coordinate without disclosing their own plans, and then decide to accept or not these tasks. Looking at this problem in a more generic way, we consider a set of agents having their own private slots on some disjunctive resources, within which they can schedule whatever they want. Some requests from non-owners, may require access to one or more such private slots. This problem amounts to finding an allocation of tasks to private slots, so that requests are fulfilled in the best manner, whilst minimizing their impact on the private plans and their disclosure; which can be instantiated to other application fields, such as collective robotics, autonomous cars, or UAVs.

This problem falls into the multi-robot/multi-agent task allocation frameworks [7, 18]. Among the efficient solution methods for solving such problems, market-based allocation techniques have recently received particular attention, and have been used in collective robotics [12] and Space [14, 19] domains. Here, agents (or robots) implement auction-based algorithms to allocate the tasks in a distributed cooperative manner. Agents bid on single tasks or bundle of tasks they want to insert into their plans, and winners are determined depending on the value of the bids, in a centralized (e.g. combinatorial auctions, PSI, SSI [13]) or decentralized (e.g. CBBA [2]) manner. In the domain of observation task allocation, We provided models and algorithms for Earth observation-specific scenarios with private slots [20], where each mode only contained a single task. In a slightly different direction, Phillips and Parra investigate the use of consensus-based bundle allocation between satellite themselves, but without any privacy concern [19]. Finally, Lee et al. consider several constellations owned by different agents and propose to use consensus-based protocols to allocate tasks within these constellations [14]. In all these models and algorithms, composite requests are not considered. In such approaches tasks are not interdependent or part of a composite task: agents only care about their bundle consistency and value, and not about some constraints over the set of atomic tasks from the same composite task. To meet such item interdependencies, one may rely on a centralized winner determination, guaranteeing the constraints or the criteria over composite tasks are optimized [15]. In more distributed settings, such as consensus-based task allocation (e.g. CBAA or CBBA), where the winner determination is distributed, capturing task interdependencies is not often considered. However,

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

Hunt et al. proposed an extension of CBBA, called CBGA, to handle multi-agent tasks, i.e. tasks requiring several agents to be performed [10]; but again, only one mode was considered for each such task.

We propose here an extension of CBGA to cope with multi-mode composite tasks, and apply it to an Earth observation scenario with multiple orbit slot owners. Our contributions are the following:

(1) We model the multi-agent multi-mode composite task allocation problem (MACTA);
(2) We map MACTA to a mathematical program, and propose a greedy algorithm to solve it;
(3) We devise a distributed solver for MACTA (MM-CBGA);
(4) We map an extended version of Earth observation satellite constellation scheduling problem [20] to MACTA, by considering composite tasks instead of monoscopic tasks;
(5) We assess MM-CBGA performances on realistic order books and a constellation simulator.

The paper is structured as follows. Section 2 provides the details of the MACTA problem model. Section 3 defines a mathematical program to model MACTA, and a greedy algorithm as to scale up its resolution. Section 4 presents the core contribution of the paper, MM-CBGA. Section 5 describes the experimental setup and analyzes the results obtained with MM-CBGA against a centralized baseline, and a single item sequential auction algorithm. Section 6 concludes the paper with some perspectives.

2 PROBLEM MODEL AND NOTATIONS

This section introduces the multi-agent composite task allocation problem we address in this work.

Definition 2.1 (Slot). A *slot* $w = [s_w, e_w]$ is a time window, where $s_w \in [t_{\min}, t_{\max}]$, $e_w \in [t_{\min}, t_{\max}]$, $s_w < e_w$, $t_{\min} \in \mathbb{R}_{\geq 0}$, $t_{\max} \in \mathbb{R}_{\geq 0}$, and $t_{\min} < t_{\max}$.

Let $\mathcal{T}$ be a set of tasks to perform and $\mathcal{R}$ a set of *disjunctive resources* i.e. that can only be used for a single task at a time. In our paper, tasks will be observation tasks to perform on satellites, that can only perform one observation at a time.

Definition 2.2 (Task). A *task* $\tau \in \mathcal{T}$ is a tuple $\langle w_\tau, d_\tau, r_\tau, \omega_\tau \rangle$ where $w_\tau = [s_\tau, e_\tau]$ is the slot during which the task can be scheduled, $d_\tau \in \mathbb{R}_{\geq 0}$ is the duration of the task, $r_\tau \in \mathcal{R}$ is a disjunctive resource on which the task has to be performed, and $\omega_\tau \in \mathbb{R}_{\geq 0}$ is the reward received for performing the task.

Definition 2.3 (Schedule). A *schedule* is a set $\pi = \{(\tau, t) \mid \tau \in \mathcal{T}, t \in [t_{\min}, t_{\max}]\}$ defining the start time t of each task τ, such that there is no overlap between any tasks on the same resource: $\forall (\tau, t), (\tau', t'), [t, t + d_\tau] \cap [t', t' + d_{\tau'}] = \emptyset$.

Let $\mathring{\mathcal{T}}$ be a set of requests to fulfill. In our experiments, some requests will arise from client users that can be fulfilled by performing observations on private orbit slots.

Definition 2.4 (Request). A *request* (or *composite task*) $\mathring{\tau} \in \mathring{\mathcal{T}}$ is a tuple $\langle M_{\mathring{\tau}}, \oplus_{\mathring{\tau}} \rangle$ where $M_{\mathring{\tau}} \in 2^{\mathcal{T}}$ is the set of sets of tasks allowed to fulfill the request (i.e. the set of *modes*), and $\oplus_{\mathring{\tau}} : M_{\mathring{\tau}} \to \mathbb{R}_{\geq 0}$ is an aggregation function which computes the reward received for fulfilling each mode, based on its individual task rewards.

The notion of *mode* is common in resource-constrained project scheduling (RCPSP) [3], and even used in some Earth observation scheduling [22]. It is convenient to model the interdependencies between atomic tasks to fulfill requests. The aggregation function $\oplus_{\mathring{\tau}}$ can be any monotonically increasing function such as the sum (+), or a linear function with some discount factor for any task after the first one[1]. A request is fulfilled if one of its mode is scheduled, i.e. all the tasks of one mode are scheduled.

Definition 2.5 (Mode reward). For a given mode m of a request $\mathring{\tau}$, we call the *mode reward* the aggregation of the rewards of the tasks composing the mode: $\omega_m \stackrel{\text{def}}{=} \oplus_{\mathring{\tau}}(m)$.

For a given task τ, we note $\text{request}(\tau)$ (resp. $\text{mode}(\tau)$) the request (resp. mode) τ contributes to. We also note $\text{modes}(\mathring{\tau})$ the set of all modes for request $\mathring{\tau}$, and by extension we note $\text{modes}(T) = \bigcup_{\mathring{\tau}_i \in T} \text{modes}(\mathring{\tau}_i)$.

Definition 2.6 (Schedule reward). For a given schedule π, we call the *schedule reward*, noted ω_π, the sum of the rewards of the scheduled modes:

$$\omega_\pi \stackrel{\text{def}}{=} \sum_{\substack{m \in \text{mode}(\tau) \\ \tau \in \pi}} \bigoplus_{\tau \in \pi}^{\mathring{\tau}} \omega_\tau$$

Let $\mathcal{A}$ be a set of agents able to perform tasks. In our experiments, agents will be private slot owners, that can receive requests to insert some observations in their private slots.

Definition 2.7 (Agent). An *agent* $i \in \mathcal{A}$ is a tuple $\langle O_i, \mathring{\mathcal{T}}_i^-, \mathring{\mathcal{T}}_i^+, \pi_i \rangle$ where $O_i = \{(r, w) \mid r \in \mathcal{R}, w = [s_w, e_w]\}$ is a set of private slots on some resources, $\mathring{\mathcal{T}}_i^- \subset \mathring{\mathcal{T}}$ is a set of private requests it wants to schedule within its private slots, $\mathring{\mathcal{T}}_i^+ \subset \mathring{\mathcal{T}}$ is the set of external requests that i is asked to schedule, and π_i is the agent's current schedule which assigns a start time to some of its private and some external tasks.

Note that there is no co-ownership of slots, i.e. for the same resource r, there is no overlap between any private slots. Moreover, we assume that windows for private tasks are fully included in private slots. We also note that π_i is divided into two subparts: π_i^- for private tasks, and $\pi_i^+ = \pi_i \setminus \pi_i^-$ for external tasks from the clients. We identify the tasks an agent can perform, i.e. whose time windows are included in some of its private slots, using $\text{can} : \mathcal{A} \to 2^{\mathcal{T}}$. By extension we also use can for composite tasks, if at least one task can be scheduled in one of the agent's private slot.

Definition 2.8 (MACTA). A *multi-agent multi-mode composite task allocation problem* (or MACTA) is a tuple $P = \langle \mathcal{A}, \mathcal{R}, \mathring{\mathcal{T}} \rangle$, defined by a set of agents $\mathcal{A}$, a set of disjunctive resources $\mathcal{R}$, and a set of requests $\mathring{\mathcal{T}}$ (private or not), and it amounts to finding the allocation of atomic tasks to agents, i.e. agent schedules integrating the atomic tasks, which maximizes the sum of the rewards of the fulfilled requests, whilst meeting disjunction constraints between resources:

$$\max \sum_{\mathring{\tau} \in \mathring{\mathcal{T}}} \omega_{\mathring{\tau}}$$

[1]For readability and simplicity, we will use the simple sum (+ and $\sum$), instead of the request-specific operator ($\oplus^{\mathring{\tau}}$ and $\bigoplus^{\mathring{\tau}}$) in the rest of the paper.

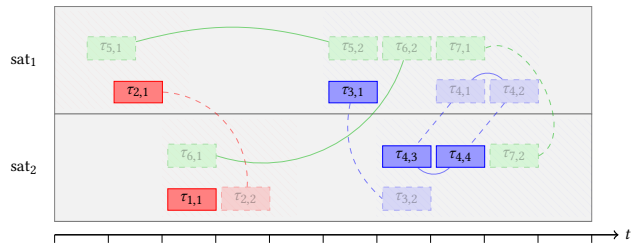

Figure 1: A simple MACTA, as described in Example 2.9. Private slots are hatched red (u_1) and blue (u_2). Currently slotted tasks are solid colored rectangles, non slotted ones are dashed. Tasks from the same request are grouped together with solid lines when tasks belong to the same mode, or dashed lines when tasks belong to different modes.

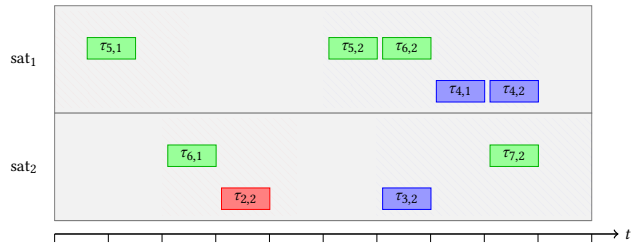

Figure 2: A solution schedule for example in Figure 1.

For a given MACTA P, we note $\mathring{\mathcal{T}}^+ \stackrel{\text{def}}{=} \bigcup_{i\in\mathcal{A}}\{\mathring{\mathcal{T}}_i^+\}$ the set of tasks which are not private to any slot owners. For a given agent i, we note P_i^- the problem restricted to $\mathring{\mathcal{T}}_i^-$, as to compute its initial schedule π_i or to revise it, given some additional tasks. As a shortcut, we note $P_i^- \uplus M$ the problem consisting in solving the private P_i^- with extra modes contained in the set M, where each mode is filtered so that only tasks within i's slots are considered. It will be used to assess the cost of slotting modes into the schedules.

Note that, while agents may have initial private schedules, we consider these schedules can be revised as to increase the collective reward. However, if some agents are reluctant to revise their schedule, they can either always answer "no" to any request overlapping their schedule, or increase the reward of their own tasks and requests to guarantee their presence in the final schedule.

Example 2.9. Figure 1 illustrates a simple MACTA. We consider the time frame $[t_{\min}, t_{\max}] = [0, 10]$, and a constellation with two satellites, $\mathcal{R} = \{\text{sat}_1, \text{sat}_2\}$. Two agents, $\mathcal{A} = \{u_1, u_2\}$, own two private slots each, emit two requests into their private slots, and already have computed some initial schedule to fulfill their own requests. For instance, u_1 emitted two requests: $\mathring{\tau}_1$ with a single mode $\{\tau_{1,1}\}$, and $\mathring{\tau}_2$ with two modes $\{\tau_{2,1}\}$ and $\{\tau_{2,2}\}$. u_2 emitted two requests: $\mathring{\tau}_3$ with two modes $\{\tau_{3,1}\}$ and $\{\tau_{3,2}\}$, and $\mathring{\tau}_4$ with two modes $\{\tau_{4,1}, \tau_{4,2}\}$ and $\{\tau_{4,3}, \tau_{4,4}\}$. Particularly, the "all-or-nothing" $\mathring{\tau}_4$ requires two tasks to be performed, or none.

We consider the following rewards for tasks: $\omega_{\tau_{1,1}} = \omega_{\tau_{2,1}} = \omega_{\tau_{4,3}} = \omega_{\tau_{4,4}} = 10$, $\omega_{\tau_{2,2}} = \omega_{\tau_{3,1}} = \omega_{\tau_{3,2}} = \omega_{\tau_{4,1}} = 8$, $\omega_{\tau_{4,2}} = 6$. Assuming the aggregation function for requests is +, the best private schedules for agents are those represented in Figure 1, selecting the best modes for each request. The resulting reward is 48.

Now, when some external requests arise, the private schedules may drastically change as to increase the global reward. We consider three new requests (in green in the figures) emitted by an external user, with all the same atomic task reward of 6: $\mathring{\tau}_5$ with one mode $\{\tau_{5,1}, \tau_{5,2}\}$, $\mathring{\tau}_6$ with one mode $\{\tau_{6,1}, \tau_{6,2}\}$, and $\mathring{\tau}_7$ with two modes $\{\tau_{7,1}\}$ and $\{\tau_{7,2}\}$. Without revising u_1 and u_2's initial schedules, the only fulfilled external request is $\mathring{\tau}_7$, by scheduling $\tau_{7,2}$, which results in a schedule reward of $48 + 6 = 54$.

u_1 benefits in removing task $\tau_{1,1}$ to schedule $\tau_{6,1}$ and $\tau_{6,2}$, for a gain of $6 + 6 - 10 = 2$, but requires coordination with u_2 who has to insert $\tau_{6,2}$ in its schedule. u_1 also switches from $\tau_{2,1}$ to $\tau_{2,2}$, leaving room for $\tau_{5,1}$, and requiring scheduling $\tau_{5,2}$, which obliges u_2 to switch from $\tau_{3,1}$ to $\tau_{3,2}$, which in the end generates a gain of $6 + 6 - 10 + 8 - 8 = 2$. The schedule reward of the solution illustrated in Figure 2, which is optimal, is 60. However, achieving such quality requires the agents to coordinate when requests have different modes concerning several private slots.

Finally, note that for simplicity, we did not consider tasks can be positioned in flexible time windows: their time windows are single points. Thus, other schedule revisions and adaptations would have been possible with larger margins, by sliding some tasks.

3 CENTRALIZED APPROACHES

A classical approach to centrally solve an allocation problem consists in using a mathematical program –ideally a linear program, as to benefit from efficient off-the-shelves solvers, such as IBM CPLEX [11] or Gurobi [9]. Alternatively, one may rely on greedy algorithms, as to scale up. Notice that these solutions are clearly not respectful of privacy: agents would have to send all their data to a central authority in charge of computing the schedules. However, they represent good baselines for assessing quality of the solutions obtained in a decentralized manner, and they can be used by agents to compute or assess private schedules, only concerning their own requests and external requests on their private slots.

Optimal Approach. We formulate our problem as a mixed integer linear program (MILP). Let's consider our decision variables: $x_\tau \in \{0, 1\}$ a binary variable stating whether task τ is scheduled; $y_m \in \{0, 1\}$ a binary variable stating whether mode m is selected for fulfilling some request $\mathring{\tau}$; $t_\tau \in [s_{w_\tau}, e_{w_\tau} - d_\tau]$ a continuous variable stating the starting time of τ; and $\beta_{\tau,\tau'} \in \{0, 1\}$ a binary variable stating whether τ precedes τ'. We note $(\tau, \tau') \in \mathcal{T}_\cap^2$ pairs of distinct tasks on the same resource with intersecting time windows.

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

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

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

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

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

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

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

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

with $\Delta^{\max}_{\tau,\tau'} = e_\tau - s_{\tau'} + d_\tau$ a value serving as a big-M constant [8] to trigger constraints conditioned by precedence between two tasks. This MILP aims at maximizing the reward from active modes, as stated in Equation (1). Constraint (2) links x's and y's by stating that if a mode is selected all its tasks should be scheduled. Constraint (3) forces to select at most one mode per request. Constraints (4) to (6) ensure the consistency of precedence variables, while Constraints (7) and (8) guarantee that distinct tasks on the same resource do not intersect. Unfortunately, such a program, given the presence of potentially numerous integer and binary variables, will not scale up. So, one would have to consider non optimal approach, such as greedy allocation, which have shown good performance on similar task allocation problems [1, 23]. We therefore sketches in the following paragraph a greedy algorithm to fulfill requests.

Greedy Approach. The greedy algorithm, presented in Algorithm 1, first sorts modes in decreasing reward order (lines 4). For each mode, it attempts to find slots for all tasks in the mode (lines 5-11). If all tasks can be scheduled, the mode is selected and all tasks are scheduled (lines 12-14). Otherwise, the successful slots are removed from the resource schedule. To find a slot, the function `first` simply searches for the first available slot on the resource schedule, consistent with the time window of the task, without any overlap with any already slotted task. This greedy algorithm is not optimal, but provides very fast solutions, since it is polynomial in the number of modes and tasks. But, as for MILP, it requires sharing all the constraints and information with a central scheduler authority. This algorithm will be our baseline for experiments, since greedy algorithms are largely used in space mission scheduling, by constellation operators, and because the MILP solver did not scale up even for the smallest problems we consider, while greedy algorithms provide solutions in less than few minutes on larger problems.

Algorithm 1: Greedy MACTA Solver

Data: A MACTA $P = \langle \mathcal{A}, \mathcal{R}, \mathring{\mathcal{T}} \rangle$
Result: A schedule π

1 $\pi \leftarrow \{\}$ // the schedule to build
2 $R \leftarrow \{(r, [])\} \mid r \in \mathcal{R}\}$ // the resource schedules
3 $S \leftarrow \{\}$ // the fulfilled requests
4 $M \leftarrow \texttt{sort}(\texttt{modes}(\mathring{\mathcal{T}}))$ // the sorted modes
5 **for** $m \in M$ **do**
6 **if** request$(m) \notin S$ **then** // request not fulfilled
7 $T \leftarrow \{\}$ // the slots found so far
8 **for** $\tau \in m$ **do**
9 $t \leftarrow \texttt{first}(\tau, P, R)$
10 **if** $t \neq \emptyset$ **then** // slot found for task
11 $T \leftarrow T \cup \{t\}$
12 **if** $|T| = |m|$ **then** // all slots found for mode
13 $\pi \leftarrow \pi \cup \{(\tau, t) \mid t \in T\}$
14 $S \leftarrow S \cup \{\text{request}(m)\}$
15 **else** // no room for all tasks
16 $\texttt{remove}(T, R)$ // remove explored slots
17 **return** π

4 MULTI-MODE CONSENSUS-BASED BUNDLE ALLOCATION FOR COMPOSITE TASKS

This section exposes the core contribution of the paper: MM-CBGA, a consensus-based algorithm for allocating composite tasks with multiple modes to a set of agents. Beforehand, we briefly recall some background on auction-based task allocation.

4.1 Auction-based Task Allocation

The classical task allocation framework consists in a set of resources and a set of tasks to be performed on some resources. The objective is to assign tasks to resources so that this optimizes some criteria (e.g. the number of assigned tasks, the sum of the rewards of the slotted tasks, the makespan of the overall schedule, etc.). This is a classical allocation problem that can be modeled as a MILP (similar to the one in the previous section). The idea is that the tasks to be scheduled are open for bidding by an *auctioneer*. The *bidder* agents evaluate the tasks depending on their current plan, and send their bids for some of these tasks. Then the auctioneer determines the winners depending on their bids and on constraints on the resources. Here, the most computationally expensive operations are the bidding step performed by each bidder, which can have an exponential number of bundles to valuate, and the winner determination problem (WDP) performed by the auctioneer, which amounts to solving an Integer Linear Program with a potentially exponential size, and falls into the combinatorial auction (CA) framework [4].

Looking at the literature on multi-robot task allocation [5, 12] and multi-satellite observation allocation [14, 19, 20], these computational limits can be overcome using the classical relaxation consisting in only allowing bidding on single items (and not on bundles). When bidders bid on the whole set of items in parallel, we fall into Parallel Single Item (PSI) framework [13]. When the auctioneer announces items one by one, and bidders build their bid knowing the previous item allocations, we fall into the Sequential Single Item (SSI) framework [13]. In general SSI has very good performances with very limited computation time, while PSI solution quality is 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 (in rounds) than SSI while yielding similar solutions and having the benefits of traditional consensus algorithms [2]. CBBA is a fully distributed solution to implement a computationally cheap variant of combinatorial auctions. It follows a repeated 2-phase sequence. First, during the *bidding phase*, each agent 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 plan. Then during the *consensus phase*, the agents compare their bids with their teammates' bids. If an agent 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 has been extensively studied and modified to improve its performances and adapt it to specific scenarios, like multi-satellite observation allocation [14, 19, 20]. The very extension we focus on in this paper is CBGA, Consensus-based Group Auction [10]. Here, some multi-agent tasks require the participation of several agents

Algorithm 2: MM-CBGA solver

```
Data: A MACTA P = ⟨A, R, T̊⟩
Result: A schedule π⁺ for external requests, and a private schedule
        π_i⁻ for each agent i ∈ A
1  concurrent for each i ∈ A do
2      π_i⁻ ← solve(P_i)
3      N_i ← {j ∈ A | j ≠ i, can(j) ∩ can(i) ≠ ∅}
4  while conflict do
5      concurrent for each i ∈ A do                    // Bidding Phase
6          b_i, c_i ← bid(i)                           // see Alg. 3
7          for each j ∈ N_i do send(⟨b_i, c_i, s_i⟩, j)
8      concurrent for each i ∈ A do                    // Consensus Phase
9          for each received b_j do
10             consensus(b_i, c_i, b_j, c_j)           // see Alg. 4
11 concurrent for each i ∈ A do send(π_i⁺, client)
12 return ⋃_{i∈A} π_i⁺
```

to be completed. When a sufficient number of agents bid for a task, it can be added to bundles. To do so, CBGA differs from CBBA in two main points: (i) bid data structures store which agents bid for each task, instead of only storing the best bid value so far, and (ii) during the consensus, task rewards are computed on the basis of the sum the best agent bids to perform the task, discarding the worst ones when more agents then necessary have bid. However, neither CBBA nor CBGA can cope with our very problem, because MACTA requires some composite tasks to be performed in different manners (modes). We thus extend CBGA to multi-mode composite tasks setting, in the following section.

4.2 MM-CBGA

MM-CBGA follows the same rationale than CBBA and CBGA, as sketched in Algorithm 2. First, agents solve their private problems, using any scheduler (line 2) and connect to agents that can slot tasks from the same requests (line 3). Then, each agent concurrently bids over the set of modes, according to previously received bids, and sends the results to neighbors (lines 5-7). Next, each agent solves the conflicts it may have with bids from its neighbors (lines 8-10). When there is no more conflict between agents, allocations for external tasks are sent back to the client (line 11).

4.2.1 Data Structures. The main data structures used by each agent i are *bids* and *contributions*. A *bid* is a 3-dimensional structure where each cell contains the reward for a given request $\mathring{\tau}$, a given mode m and a given agent j in $\mathcal{N}_i$, noted $b_i[\mathring{\tau}][m][j]$. The value of a bid cell can be either (i) a finite positive or negative value, when the agent knows or assesses how much it gains or it looses in scheduling the tasks of the mode in its slots; (ii) the negative infinity, if it cannot insert the tasks of the mode in its schedule, due to resource limitation (no more room in its slots); (iii) or empty, if it has not bid yet on the mode. The bids are updated at each bidding phase, as explained later on. Complementary to a bid, a *contribution* is the set of tasks agent j is ready to schedule for given request $\mathring{\tau}$, and mode m, noted $c_i[\mathring{\tau}][m][j]$. To each bid corresponds a contribution, i.e. the set of tasks whose insertion results in the proposed bid. As to avoid deadlocks due to obsolete information, each agent i stores the date of the last message it has received from each other agent j, noted $s_i[j]$. These timestamps will also be exchanged as to avoid making decisions based on obsolete information. Finally, each agent i keeps track of its *bundle*, β_i, i.e. the set of external modes currently slotted in its schedule.

4.2.2 Bidding Phase. Differently from CBBA and CBGA, where agents bid for tasks, the idea in MM-CBGA is that agents bid on modes for each request they are aware of. Algorithm 2 presents how each agent implements this phase. The agent attempts to integrate modes in its bundle while its slots are not full (line 4). For each mode of each non fulfilled request (lines 6-7), the agent computes two schedules –with and without the mode–, to assess the marginal gain (or cost) of integrating the given mode into its current slots, according to its current bundle β_i. The resulting gain and respective slotted tasks will constitute the bid and the contribution of the agent (lines 11-12). The two schedules are computed using any scheduler, noted `solve`, that can be the previously presented MILP or greedy algorithm. Of course, the agent does not have to compute again the solution for the very same subproblem, by storing the already computed solution of each subproblem. The best mode for each request must be chosen, but when an agent has no knowledge about the bids of other agents required for a mode, it will assess an *upper bound* for the mode integration (line 13). This upper bound consists in the aggregation of the effective rewards of the existing bids, as in Equation (9) and the maximum hypothetical rewards for the missing bids on some tasks of the mode, as in Equation (10). As soon as an agent disagrees to select the mode ($-\infty$ bid), the mode is discarded for integration.

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

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

The reason for using hypothetical rewards and not only effective ones is to allow modes with negative effective reward to be considered, if they have the potential to bring some gain thank to not already discovered rewards, as illustrated in Example 2.9. In case this upper bound is over confident, it will be refined during the consensus phase, by updating the reward with effective bids. In case multiple agents can perform the very same task for a mode, and thus there is more bids on m for $\mathring{\tau}$, only the best ones, sufficient to fulfill $\mathring{\tau}$, are used in the definition of UB (the worst ones are ignored).

This upper bound is used to determine the next best mode candidate for integration to the bundle (lines 14-15). Once all the remaining requests and modes have been processed, if a best mode have been found –i.e. there is still room in the slots– and if this mode is valid, it is inserted into the bundle (lines 20-22). A mode is valid when all the tasks have been effectively chosen (line 20), and its overall reward is positive (line 21). At the end of the bidding phase, each agents will have built its bundle, bids and contributions. The two latter are then sent to the neighboring agents with the timestamps, to reach a consensus.

4.2.3 Consensus Phase. Algorithm 4 sketches how agents solve conflicts to align their bids. When agents' bids disagree on the best mode for a given request (lines 4-6), they solve the conflict as

Algorithm 3: The bid function

```
1  Function bid(i)
2      M ← ∅                                              // already analyzed modes
3      R ← ∅                                              // already analyzed requests
4      while true do
5          ω* ← −∞, τ̊* ← ∅, m* ← ∅
6          for each τ̊ ∈ T̊_i^+ \ R do                      // requests
7              for each m ∈ M_τ̊ \ β_i do                  // modes
8                  π_i ← solve(P_i^− ⊎ β_i)
9                  π_i^m ← solve(P_i^− ⊎ (β_i ∪ {m}))
10                 if m ∈ {mode(τ) | (τ,t) ∈ π_i^m} then
                       // m's tasks in i's slots
11                     c_i[τ̊][m][i] ← {τ | (τ,t) ∈ π_i^m}
                       // assess marginal gain
12                     b_i[τ̊][m][i] ← ω_{π_i^m} − ω_{π_i}
13                     ω ← UB(b_i[τ̊][m])
14                     if ω > ω* then                     // best mode so far
15                         ω* ← ω, τ̊* ← τ̊, m* ← m
16                 else                                   // m cannot be scheduled
17                     b_i[τ̊][m][i] ← −∞
18                     c_i[τ̊][m][i] ← ∅
19         if ω* ≠ −∞ then                                // a mode has been found
20             if m* ⊆ ⋃_j c_i[τ̊*][m*][j] then             // the mode is full
21                 if Σ_j b_i[τ̊*][m*][j] > 0 then
22                     β_i ← β_i ∪ {m*}                   // integrate to bundle
23             M ← M ∪ {m*}                               // do not consider m* again
24             R ← {τ̊ | M_τ̊ ⊆ M}                          // do not consider full τ̊
25         else break
26     return b_i, c_i
```

Algorithm 4: The consensus function

```
1  Function consensus(b_i, c_i, b_j, c_j)
2      conflict ← false
3      for each τ̊ ∈ T̊_i^+ ∩ T̊_j^+ do                      // shared requests
4          m_i ← arg max(b_i[τ̊])
5          m_j ← arg max(b_j[τ̊])
6          if b_i[τ̊][m_i] ≠ b_j[τ̊][m_j] then              // conflict
7              conflict ← true
8              for each m ∈ M_τ̊ do                        // update with up-to-date bids
9                  for each k ∈ b_i[τ̊][m][·] do
10                     if b_j[τ̊][m][k] ≠ ∅ then
11                         if k ≠ i and s_j[k] > s_i[k] then
12                             b_i[τ̊][m][k] ← b_j[τ̊][m][k]
13                             c_i[τ̊][m][k] ← c_j[τ̊][m][k]
14             m_i* ← arg max_{m∈M_τ̊} {UB(b_i[τ̊][m])}
15             if m_i* ≠ m_i and m_i ∈ β_i then            // new best mode
16                 discard(β_i, m_i)                      // remove m_i and followers
```

in CBGA, i.e. an agent updates its bids when receiving bids from neighbors with fresher information (lines 8-13). Depending on these updated bids and the possible change of mind about the best mode, an agent may suppress modes from its bundle (lines 14-16). The function `discard` removes from the bundle a mode and all the modes inserted after this mode, as to reconsider the task allocation.

4.2.4 Convergence. In order to converge, consensus-based algorithms such as CBBA and CBGA require that the reward of inserting a task into a schedule can't increase if other tasks are inserted before it. Here this condition translate into : $\texttt{solve}(P_i^- \uplus \beta_i) \leq \texttt{solve}(P_i^- \uplus (\beta_i \cup \{m\}))$. This condition, called as *diminishing marginal gain* [2], is satisfied in MM-CBGA since we always schedule each task of each bundled mode at the time that yields the maximum score increase. So, if we tried inserting the same tasks into a fuller schedule, then there would be fewer time windows available and the task would either need to be placed at a less optimal time or at the same time as in the original schedule if that time window still does not overlap any other task, as described in [19].

4.3 Main Differences with CBGA

MM-CBGA differs from CBGA in several ways. (i) MM-CBGA does not specifically focus on agents having path to follow to perform tasks. In CBGA, it has a great impact on the way rewards are assessed, using distance and time to fulfill tasks. MM-CBGA adopts a more abstract vision and makes use of an upper bound calculation to discard unfeasible modes, and to keep still promising modes. (ii) In MM-CBGA agents own several finite slots where they can schedule tasks, while in CBGA agent have an infinite task queue. (iii) In MM-CBGA agents bid on modes for multi-mode composite tasks, while CBGA agents bid on composite tasks with unique mode. Data structures have been extended in MM-CBGA as to keep track of the tasks performed by each agent to detect mode completion, using so-called *contributions*. As a result, the bidding phase is slightly more complex than CBGA. But let's note that on mono-mode settings MM-CBGA is equivalent to CBGA. (iv) In return the consensus phase is simpler, since mode completion is fully integrated to mode reward assessment with `UB`. In the mono-mode case, MM-CBGA is equivalent to CBGA, but instead of determining which agents win a request in the consensus phase, this information is derived from the bids themselves by computing `UB`.

5 EXPERIMENTAL EVALUATION

This section presents the experiments we conducted to evaluate the performances of MM-CBGA. They are coded in Java 11.0.15 and executed on 20-core Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz, 62GB RAM, Ubuntu 18.04.5 LTS. We evaluate the performances of MM-CBGA compared to two baselines: a centralized solver (centralized, which is greedy in our case) that provides good quality (close to optimal) solutions in very limited time, and a sequential single item auction (ssi) which provides solutions with quality equivalent to CBGA in mono-mode settings. The `solve` procedure used in MM-CBGA, and ssi is also the greedy algorithm presented in Section 3. The computation time reported latter is a mono-CPU computation time (no distributed execution).

5.1 Earth Observation Scenario

The scenario we use to evaluate MM-CBGA is strongly inspired by EOSCSP (Earth Observation Satellite Constellation Scheduling Problems) [20], which initially consists in a multi-mode non-composite

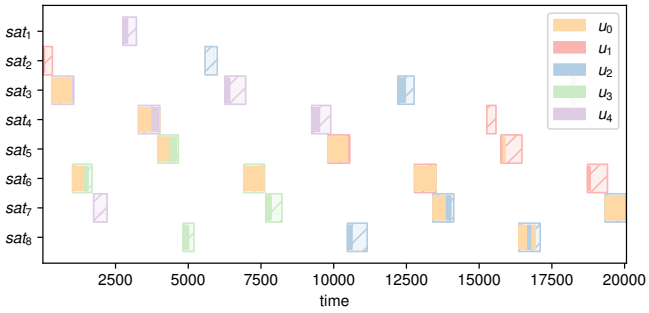

Figure 3: The private workload schedules for a constellation with 8 satellites and 4 slot owners (u_1 to u_4) and a client (u_0) having 83 requests fulfilled with 275 tasks in the private slots to observe 10 European cities, over 6 hours.

MACTA (each request is fulfilled by exactly one tasks among several possible ones). Thus, we generated MACTA instances representing observation task scheduling over a set of satellites with several orbit slot ownerships.

We consider a Low-Earth Orbit constellation (500km altitude). The constellation is composed of 4 orbital planes with a 45 degrees inclination. Each orbital plan contains two satellites in phase opposition, for a total of 8 satellites. This constellation setting is used for determining the orbit slot ownerships and the task time windows according to some points of interest (POI) on Earth, using a dedicated spatial mechanics library. The scenario we simulate spreads over 6 hours (21600 seconds).

4 users (our agents, u_1 to u_4) own up to 10 orbit slots each, and will attempt to slot requests emitted by a fifth user (u_0). Orbit slot ownerships are visibility windows to some randomly chosen POIs (10 amongst 27 European cities). Orbit slots are assigned using a round-robin procedure: each user chooses the best remaining visibility window to observe one of its POIs, and let the next user choose its next orbit slot, until there is no more window, or the limit of 10 ownerships per user is reached. On average, on the all instances we considered, according to the maximum incidence angle ($\theta_{\max} = \frac{\pi}{6}$) for performing good quality observations, the orbit slot ownership duration is approximately 10 minutes (595.03 seconds). Figure 3 presents one such scenario.

All users emit some requests. This number will vary over the experiments to evaluate how it impacts the performances of MM-CBGA. Orbit slot owners emit 2 to 20 requests in their own slots, whilst the non-owner user emits 4 to 80 requests (as many as all the other users) in exclusive slots, to stress the system. A request is defined by a POI, and thus all the existing visibility windows to acquire this POI are potential time windows for observation tasks. We studied two configurations. The first one consider only one mode per request, which consists in making 5 distinct observations of the same POIs, in the 5 best visibility windows, according to the observation angle. Thus, we fall in the mono-mode composite-task settings, where MM-CBGA is equivalent to CBGA. In the second configuration, we consider 5 modes per request, which consists in degrading the previously defined mode (5 observations) by removing 1 to 4 windows. This results in 5 modes with respectively 5, 4, 3, 2 and 1 task per mode. Task time windows are the same than the visibility windows: agent can position observations anywhere in the orbit slots. The duration of the tasks is randomly determined in $[20, 40]$ seconds. The scheduler has also to respect an inter-task satellite reconfiguration time of 10 seconds, since the satellites can only perform one observation at a time, and have to rotate to point to next POI. The incidence angle (θ), which impacts the quality of observations, is used to determine the reward of each the task τ, the closer to 0° the better: $\omega_\tau = \rho(1 - \frac{|\theta|}{\theta_{\max}})$, with ρ a random number in $[1, 2]$. This results in average rewards of approximately 1.41 (min $\approx$ 0.89, max $\approx$ 1.99).

We ran 100 instances of randomly generated MACTA instances with seed in [0:99] for each setting, and plot the average, with [0.05, 0.95] confidence. Random values are uniformly chosen within provided intervals. This results in 3000 instances in total, available online [21].

5.2 Performance Analysis

To assess MM-CBGA performances, we analyze the metrics from Figure 4, for two configurations: 1 mode per request (top row) and 5 modes per request (bottom row). On each row, we plot the quality ratio compared to the centralized greedy solver, the percentage of fulfilled requests, the number of slotted tasks, the number of exchanged messages, the communication load (in MB), and the total computation time (in s), with an increasing number of requests.

Mono-mode Requests. We first look at the results on MACTA with a single mode per request (top row of Figure 4). In this situation, the problems to solve are equivalent to those solved using CBGA. As expected the quality of MM-CBGA's solutions are equivalent to a sequential single item approach (ssi). Since there is only one mode per request, consisting in five observation tasks, centralized cannot find solutions fulfilling all the requests. Moreover, with numerous requests (more than 160), on this limited resource system, centralized is even outperformed by MM-CBGA in terms of requests (and thus of tasks), because this latter allows requests with non immediate reward to be scheduled. Looking at the operational metrics concerning communication and computation, because the network between agents is stable, MM-CBGA only requires a small constant number of messages, while ssi requires a number of messages linear in the number of modes. However, the data structures exchanged in MM-CBGA are quadratic, which results in similar communication load between MM-CBGA and ssi. The quantity of information is also tracked for centralized: indeed, slot owners need to send all their data to the solver. Computation-wise, MM-CBGA, which requires building and aggregating larger data (modes times agents), is two orders of magnitude slower than centralized. Thus, in this mono-mode setting, MM-CBGA behaves very similarly to ssi in terms of quality, communication and computation. Indeed, it is equivalent to CBGA. But, contrary to ssi, MM-CBGA is fully decentralized: the WDP is distributed among all the agents.

Multi-mode Requests. In this setting (bottom row of Figure 4), each request can be fulfilled by 5 modes with degradation from 5 tasks to 1 task. Thus, there is room for relaxation to fulfill requests. This is illustrated by the percentage of fulfilled requests, which does not drop as fast as in the mono-mode setting. More

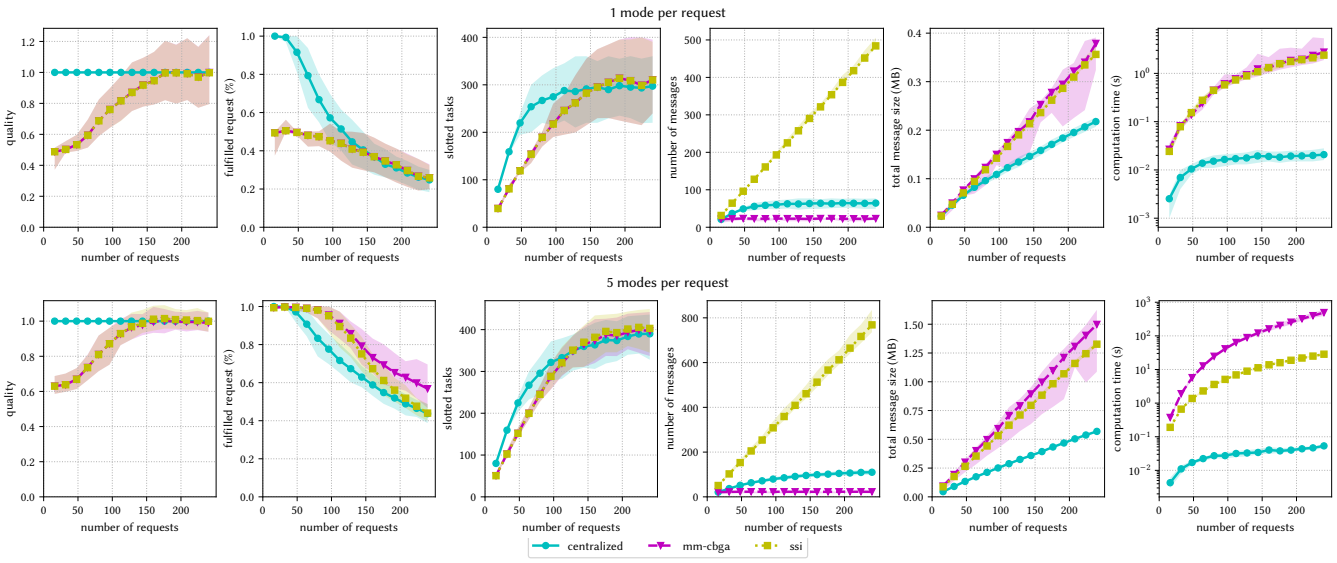

Figure 4: Performances for two configurations: 1 mode per request (top row) and 5 modes per request (bottom row).

interestingly, MM-CBGA and ssi fulfill more requests than centralized, for a lower quality on smaller problems (< 150 requests). Indeed, both auction-based approaches slot more modes with fewer tasks to fulfill more requests. On larger instances (> 150 requests), the quality and fulfillment curves of ssi and centralized converge, to reach equivalent performances, while MM-CBGA still serves more requests for an equivalent overall reward, by the use of UB. Communication-wise, the message size increases compared to the mono-mode setting for MM-CBGA, due to the presence of multiple modes in bids. For the same reason, MM-CBGA requires more time to compute bids and to solve conflicts, resulting in a higher computation time (3 orders of magnitude higher than centralized, and 1 order of magnitude than ssi). Once again, recall that MM-CBGA is fully decentralized and that we report mono-CPU time. To sum up, in multi-mode setting, MM-CBGA tends to fulfill more requests with smaller modes than centralized, which even provide results with reward equivalent to centralized on larger instances. The cost of decentralization, incremental bundle building and privacy is a lower quality on easier/smaller instances, and higher communication and computation load, in general.

6 CONCLUSIONS

In this paper we have addressed the problem of allocating composite tasks (or requests) requiring several atomic tasks to be fulfilled and having several alternatives (or modes) to be performed on private slots owned by cooperative agents. We first modeled this allocation problem (MACTA) and proposed centralized approaches to find optimal (MILP) or suboptimal (greedy) solutions. But, both suffer from their centralization and the disclosure of agents' schedules in private slots. We thus proposed a novel algorithm (MM-CBGA) to solve MACTA in a decentralized fashion, where agents coordinate to reach a consensus on bids, to decide which mode to choose for each request, while maximizing the overall reward.

We evaluated the performance of MM-CBGA on randomly generated Earth observation scheduling problems (EOSCSP), using a simulation with realistic constellation settings to determine the private slots, the modes and the tasks to observe some cities over Europe. MM-CBGA displays performances equivalent to Sequential Single Item Auction (SSI) on both single and multi-mode settings, and reaches the same quality than a centralized greedy solver on larger and harder instances. Computation-wise, MM-CBGA requires less steps to converge but more time than SSI on larger instances. This computation time remains reasonable (below 5 minutes), while there is still room for improvement by parallelizing agent operations. Communication-wise, MM-CBGA requires less, but larger messages, due to the extra information required to solve conflicts between bids. All in all, MM-CBGA turns out to be a interesting addition to the family of consensus-based algorithms, filling the gap in terms of multi-mode and multi-agent tasks.

Yet, we identify potential performance improvements, notably by devising better heuristics and upper bounds to consider requests and modes, that could be devised for specific kind of tasks (as in CBGA) or modes (e.g. modes included in others). Moreover, there exist other techniques to coordinate agent actions, e.g. DCOPs [6]. We used such algorithms, with equivalent performances to CBBA on multi-mode atomic tasks settings [20]. It could be relevant to evaluate their performances on MACTA. Finally, we evaluate MM-CBGA on static problem settings, while consensus-based algorithms are fitted for dynamic settings, where requests and agents can appear or disappear. We will investigate MM-CBGA performances with online dynamic order books, with unpredictable events due to weather conditions than can discard some tasks due to cloud coverage.

REFERENCES

[1] Doo-Hyun Cho, Jun-Hong Kim, Han-Lim Choi, and Jaemyung Ahn. 2018. Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation. *Journal of Aerospace Information Systems* 15, 11 (2018), 611–626. https://doi.org/10.2514/1.I010620 arXiv:https://doi.org/10.2514/1.I010620

[2] Han-Lim Choi, Luc Brunet, and Jonathan P. How. 2009. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Trans. Robotics* 25, 4 (2009), 912–926. https://doi.org/10.1109/TRO.2009.2022423

[3] José Coelho and Mario Vanhoucke. 2011. Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research* 213, 1 (2011), 73–82. https://doi.org/10.1016/j.ejor.2011.03.019

[4] Peter Cramton, Yoav Shoham, and Richard Steinberg (Eds.). 2010. *"Combinatorial Auctions"*. MIT Press, Boston.

[5] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. 2006. Market-Based Multirobot Coordination: A Survey and Analysis. *Proc. IEEE* 94, 7 (2006), 1257–1270. https://doi.org/10.1109/JPROC.2006.876939

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

[7] Brian P. Gerkey and Maja J. Matarić. 2004. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research* 23, 9 (2004), 939–954. https://doi.org/10.1177/0278364904045564

[8] Igor Griva, Stephen G. Nash, and Ariela Sofer. 2008. *Linear and Nonlinear Optimization*. SIAM, Philadelphia.

[9] Gurobi Optimization. 2022. Gurobi Optimizer. https://www.gurobi.com/lp/or/gurobi-optimizer/

[10] Simon Hunt, Qinggang Meng, Chris J. Hinde, and Tingwen Huang. 2014. A Consensus-Based Grouping Algorithm for Multi-agent Cooperative Task Allocation with Complex Requirements. *Cogn. Comput.* 6, 3 (2014), 338–350. https://doi.org/10.1007/s12559-014-9265-0

[11] IBM Corporation. 2022. IBM ILOG CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer

[12] Alaa M. Khamis, Ahmed Hussein, and Ahmed M. Elmogy. 2015. Multi-robot Task Allocation: A Review of the State-of-the-Art. In *Cooperative Robots and Sensor Networks 2015*, Anis Koubâa and J. Ramiro Martinez de Dios (Eds.). Studies in Computational Intelligence, Vol. 604. Springer, Cham, CH, 31–51. https://doi.org/10.1007/978-3-319-18299-5_2

[13] Sven Koenig, Craig A. Tovey, Michail G. Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J. Kleywegt, Adam Meyerson, and Sonal Jain. 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*. AAAI Press, Washington, WA, 1625–1629. http://www.aaai.org/Library/AAAI/2006/aaai06-266.php

[14] Minjoon Lee, Sung Jun Kim, Ho-Yeon Kim, and Han-Lim Choi. 2021. Consensus-based Task Scheduling Algorithm for Agile Earth Observation Satellites with Different Authorities. In *ASCEND 2021*. AIAA, Reston, VA. https://doi.org/10.2514/6.2021-4122

[15] Efrat Manisterski, Esther David, Sarit Kraus, and Nicholas R. Jennings. 2006. Forming Efficient Agent Groups for Completing Complex Tasks. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems* (Hakodate, Japan) *(AAMAS '06)*. Association for Computing Machinery, New York, NY, USA, 834–841. https://doi.org/10.1145/1160633.1160784

[16] Sara Maqrot, Stéphanie Roussel, Gauthier Picard, and Cédric Pralet. 2022. Bundle Allocation with Conflicting Preferences Represented as Weighted Directed Acyclic Graphs – Application to Orbit Slot Ownership. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation, The PAAMS Collection (LNAI, Vol. 13616)*, Frank Dignum, Philippe Mathieu, Juan Manuel Corchado, and Fernando De la Prieta (Eds.). Springer, Cham, 280–293. https://doi.org/10.1007/978-3-031-18192-4_23

[17] Sara Maqrot, Stéphanie Roussel, Gauthier Picard, and Cédric Pralet. 2022. Orbit Slot Allocation in Earth Observation Constellations. In *11th Conference on Prestigious Applications of Artificial Intelligence (PAIS'22), 25 July 2022, Vienna, Austria (co-located with IJCAI-ECAI 2022) (Frontiers in Artificial Intelligence and Applications, Vol. 351)*, A. Passerini and T. Schiex (Eds.). IOS Press, Amsterdam, NL, 3–16. https://doi.org/10.3233/FAIA220061

[18] Nathan Michael, Michael M. Zavlanos, Vijay Kumar, and George J. Pappas. 2008. Distributed multi-robot task assignment and formation control. In *2008 IEEE International Conference on Robotics and Automation*. IEEE, New York, NY, 128–133. https://doi.org/10.1109/ROBOT.2008.4543197

[19] Sean Phillips and Fernando Parra. 2021. A Case Study on Auction-Based Task Allocation Algorithms in Multi-Satellite Systems. In *AIAA Scitech 2021 Forum*. AIAA, Reston, VA, 16 pages. https://doi.org/10.2514/6.2021-0185

[20] Gauthier Picard. 2022. Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions. In *21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, Piotr Faliszewski, Viviana Mascardi, Catherine Pelachaud, and Matthew E. Taylor (Eds.). International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), Richland, SC, 1056–1064. https://doi.org/10.5555/3535850.3535968

[21] Gauthier Picard. 2023. *Multi-agent multi-mode composite task allocation problem (MACTA) instances*. Zenodo. https://doi.org/10.5281/zenodo.7550677 [Dataset].

[22] Samuel Squillaci, Stéphanie Roussel, and Cédric Pralet. 2022. Parallel Scheduling of Complex Requests for a Constellation of Earth Observing Satellites. In *PAIS 2022 - 11th Conference on Prestigious Applications of Artificial Intelligence, 25 July 2022, Vienna, Austria (co-located with IJCAI-ECAI 2022) (Frontiers in Artificial Intelligence and Applications, Vol. 351)*, Andrea Passerini and Thomas Schiex (Eds.). IOS Press, Amsterdam, NL, 100–113. https://doi.org/10.3233/FAIA220068

[23] Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. 2020. Agile Earth observation satellite scheduling over 20 years: formulations, methods and future directions. *CoRR* abs/2003.06169 (2020), 36. arXiv:2003.06169 https://arxiv.org/abs/2003.06169

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 LiChIE project (Lion Chaine Image Elargie), coordinated by Airbus Defence and Space.