Received: 11 March 2017 | Revised: 7 January 2018 | Accepted: 27 February 2018

DOI: 10.1002/sys.21433

ORIGINAL ARTICLE

WILEY

Agent-based simulation framework and consensus algorithm for observing systems with adaptive modularity

Ximo Gallud | **Daniel Selva**

Cornell University, Ithaca, NY, USA

Correspondence
Ximo Gallud, Department of Mechanical and Aerospace Engineering, Cornell University, 416 Upson Hall, Ithaca, NY 14850.
Email: ximogc@mit.edu

Abstract

In this era of the "big data revolution," the desired capabilities of Earth Observing Systems are growing fast: we need ever more frequent data sets, covering a larger part of the frequency spectrum, with lower latency, and higher spatial resolution. To better address these needs, the space systems community has been exploring the value of shifting from highly monolithic architectures, in which large and isolated spacecraft carry multiple instruments with synergistic and complementary goals, toward more distributed architectures, where the functions of these large systems are partitioned into a larger number of smaller satellites. In this paper, we present an agent-based simulation framework that can help systems engineers assess whether or not it makes sense to be able to "change system modularity during operations" by means of temporary coalitions. Systems of observing autonomous vehicles work together to perform a set of observational tasks. The vehicles can decide to form physical coalitions with other vehicles for collective sensing of a target, when no agent alone can carry out the task, or individual observation results in degraded satisfaction. The framework extends the well-known decentralized Coupled-Constraint Consensus-Based Bundle Algorithm to multivehicle single-task allocation and introduces constraints on the formation of coalitions, so that agents can create or split a coalition depending on the benefits and costs associated with these actions. The framework is described in detail and demonstrated on a case study.

KEYWORDS

coalition formation, consensus algorithms, distributed satellite systems, multiagent systems, SEE25 systems of systems (SoS)

1 | INTRODUCTION

1.1 | Background and motivation

Earth Observing Systems (EOS) are extremely important to society, as they provide measurements of the Earth's land, ocean, and atmosphere that help advance science including atmospheric science, hydrology, or oceanography, and enable societal applications such as climate monitoring, weather forecasting, precision agriculture, and water management.

The architecture of EOS has been relatively constant for the last decades. It is mostly based on a small number of large spacecraft weighing around 1–5 mt, carrying multiple instruments with synergistic and complementary goals, and limited to almost no real-time decision-making capabilities on board, as most of the observation tasks are planned in the ground and sent to the spacecraft via ground-to-space links. This strategy is interesting from the science perspective, as measurement sets from different instruments can easily be cross-registered, and it may also be advantageous in terms of cost in some cases, as only a single spacecraft has to be developed, operated, tested, and launched. However, developing these large monolithic systems has proved to be challenging due to conflicting instrument requirements and complex interfaces (e.g., electromagnetic interference, vibrations, thermal coupling). The limited nature of on-board resources such as available power and data rate has constrained duty cycles for highly demanding instruments. Moreover, current trends suggest that requirements for more frequent revisit times (subhour in some cases), lower latency (minutes), higher spatial resolution (submeter), better accuracy, and multispectral and multiangular sampling are growing even faster than technology can accommodate them, so this problem is likely to grow in the future.[1]

In part due to those issues, the EOS architecture paradigm has started to change in the last few years. Distributed architectures in which functionality is allocated to a number of smaller components are being explored due to their advantages in terms of risk, flexibility, scalability, and evolvability.[2,3] An example of the rise of distributed

EOS can be found in the increasing number of constellations of small Earth observing satellites such as CubeSats: NASA is currently developing at least two constellations of smallsats, namely CYGNSS and TROPICS, and there are several other constellations already in place or in development in the commercial world, such as Planet, Spire, and OmniEarth. Such constellations are interesting because they can break the traditional trade-off between Geostationary Earth Orbit (GEO) and Low Earth Orbit (LEO) missions in an affordable way. Indeed, LEO missions have been limited by temporal resolution, despite having good spatial coverage, while GEO missions have poor spatial resolution but good temporal resolution within the area of coverage. With the aid of nanosatellites, one can launch a relatively large number of assets into LEO for the same cost as one large satellite, thanks to reductions in launch cost, exploitation of commercial-off-the-shelf components, and economies of scale. Such constellation could achieve revisit times similar to those of GEO satellites (e.g., down to 15–20 minutes[4]). Other advantages of distributed architectures include shorter time to science, lower risk of schedule slippage and cost overrun, and higher resilience and evolvability.[5] In addition to constellations, several other examples of distributed concepts for EOS have been studied and even implemented, including trains (e.g., NASA A-Train), clusters, swarms, and fractionated spacecraft.[6,7]

More recently, Federated Satellite Systems have been proposed, in which space assets owned and operated by different organizations engage in transactions of unused resources.[8] In this body of work, resources are defined as any commodity or service that could be potentially traded or shared between spacecraft such as downlink capacity, data processing power, or data storage capacity. These exchanges may improve the utilization efficiency of spacecraft resources and are necessary for the eventual creation of an in-orbit market of space commodities. While going from monolithic to distributed systems means allocating functions over multiple components, going from distributed systems to federated systems requires decentralizing operation and ownership of the assets. The monolithic versus distributed structural trade-off may be considered different from the centralized versus decentralized operational trade-off. However, Mosleh et al.[9] recently proposed a taxonomy and classification in which they argued that these trade-offs are similar in nature and can actually be studied as a single-axis modularity trade-off.

This work also intends to study trade-offs across the entire modularity axis, although it focuses primarily on higher modularity systems with decentralized decision making. Indeed, looking at the future of EOS, experts have proposed concepts such as sensor webs,[10] which are highly distributed systems-of-systems including space, air, and ground assets with decentralized sensing and decision making and opportunistic collaborations.[11,12] In all these concepts, resource sharing and cooperation is very important.

In this context, the motivation for this work is twofold: to try to improve our understanding of resource sharing and opportunistic collaborations in EOS by means of agent-based simulation, and to develop new algorithms and decision support to make optimal operational decisions about the formation of coalitions between assets. By improving our understanding of key operational aspects of this system through simulation and by proposing new algorithms for collaborative decision making, we hope to also improve the design of such systems.

Simulating and analyzing collaboration agreements between assets in a distributed EOS is challenging because it requires adequately modeling different operational aspects and translating those into estimated mission benefits and costs. Indeed, such collaboration agreements have several implications, specifically when this engagement comes as a result of *opportunistic* decision making (i.e., mutual benefit for all the parts involved). For instance, consider an example with two assets, one carrying an "infrared" sensor and the other carrying a "microwave" sensor—the specificities of these sensors are abstracted out for ease of understanding. Let us assume that they have agreed to simultaneously observe a specific target (e.g., a tropical storm, or a forest fire) in both spectral regions. In order to do that, these assets will have to:

- Adapt their schedules: The assets will have to agree to visit the specific target simultaneously or within a short time window, so that the information taken from the images is not decorrelated. Time decorrelation is characterized by the dynamics of the target being observed and may vary from seconds (e.g., cloud formation) to years (e.g., solid Earth).
- Adapt their trajectories: The assets will have to agree to modify their paths so they can also meet spatial constraints in the observation of the target (e.g., they may need to remain close in space so that the range and observation angle is similar for the two assets). There is also a spatial decorrelation aspect similar to the time decorrelation problem described above.
- Agree to spend resources on engagement: The assets may have to spend some of their resources (e.g., propellant, power) to change their paths, or in the case of "physical coalitions" through docking maneuvers.[13]

Therefore, a simulation method that explicitly considers the trade-off between the cost of satisfying these constraints and the benefits of the enhanced observation can improve decision making and ultimately the design of EOS. One of the key objectives of this work is to provide quantitative and qualitative insights about the circumstances under and extent to which resource sharers, given that they act opportunistically, may choose to spend resources on meeting these constraints to access higher rewards.

To tackle these research questions, we propose an agent-based framework for simulating a general class of observing systems, namely networks of vehicles (ground, air, or space) with variable autonomy (e.g., attitude and position control is always autonomous but guidance and scheduling may not be) that carry different types of sensors (e.g., cameras, spectrometers) as well as radios to communicate with each other and with a mission operations center (MOC).

In this framework, the MOC defines tasks to observe a given target in one or more spectral regions (e.g., microwave, infrared, visible) with a certain set of requirements (e.g., revisit time, spatial resolution) and a score value, which may change with time. The goal of the system is to simulate the allocation and implementation of the observational tasks

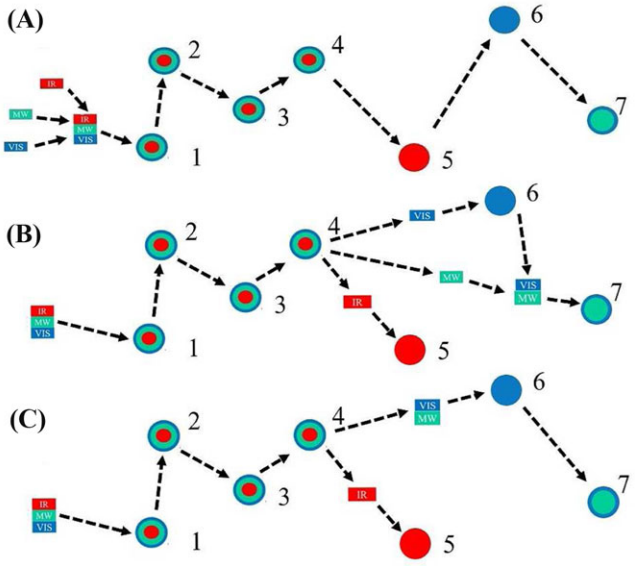

FIGURE 1 Illustration of three different plans for a scenario with three single-sensor agents and seven tasks

while trying to maximize the individual utilities of the agents (decentralized approach).

Importantly, these distributed systems can form temporary "physical coalitions," that is, they can establish a physical interface when needed through which they can share resources with each other and thus essentially become a larger multisensor system for a certain period of time (see Figure 1). The cost invested by the agents on building the coalitions (which we call *engagement cost*) is considered in this work. Figure 1 illustrates this idea of forming coalitions in an example scenario with seven tasks and three single-sensor agents. Tasks 1, 2, 3, and 4 require three different sensors: infrared (IR, represented by a red color in the figure), microwave (MW, in green), and visible (VIS, in blue). Tasks 5 and 6 require just one sensor each, and task 7 requires two sensors. Figure 1 A shows a plan in which the three single-sensor vehicles form a coalition at the beginning and perform the seven tasks as a coalition. Figure 1 B shows a plan in which agents start coalesced and the union splits after completing task 4. Tasks 5 and 6 are performed by the original single-sensor vehicles, and agents carrying the MW and VIS sensors join again for addressing task 7 together. Figure 1 C shows a path in which only the IR sensor leaves the coalition, and the VIS and MW sensors stay united to perform tasks 6 and 7.

The property of adapting the configuration of a vehicle (specifically, changing from monolithic to distributed systems or viceversa), as required by the specific properties of a given scenario including the distribution of different types of tasks, is what this paper calls *adaptive modularity*, following the taxonomy defined in Ref. 14.

Another important aspect of this framework is the score structure of the tasks. If the conditions to fully satisfy the task are not met, agents gain partial scores consistent with the level of degradation. For example, if a vehicle *V* cannot find a coalition mate to do a task that requires more sensors than *V* can provide, *V* can still do the task and obtain a lower score. We are particularly interested in scenarios with *superadditive* score structure, in which the score of a task involving sensors *A* and *B* is higher than the sum of the degraded scores allocated to performing the task with either just *A* or just *B*. The superadditive score structure makes the formation of coalitions interesting. Otherwise, optimal plans may not include any coalitions. We believe that this assumption is realistic for many classes of modern observing tasks, due to the increasing value and popularity of data fusion and cross-registration of multispectral data sets.

In an effort to keep the model as simple as possible, realistic orbital dynamics were left out of the scope of this paper, whose focus has been to explore decentralization and coalition formation in a networked system similar to Federated Satellites. A future paper will be more realistic and incorporate orbital dynamics and other aspects specific to space vehicles.

1.2 | Literature review

Formally, the coalitional formation problem we are trying to solve in this paper is described as follows using nomenclature in Ref. 15. Consider a set of agents $\boldsymbol{A} = \{a_1, a_2, a_3, \dots, a_n\}$ and a set of tasks $\boldsymbol{V} = \{v_1, v_2, v_3, \dots, v_m\}$. Lets consider $\tau_t^{a\to v}$ as the allocation of task v to an agent a at a time t.

Let us define $\Phi = \{\tau_t^{a\to v}\}_{a\in A, v\in V, 0\le t\le t_{\max}}$ as the set of possible allocations of agents to tasks in a time $t < t_{\max}$. Agents working on the same task will be considered part of a coalition $C \in 2^A$ (i.e., the power set of A). We define $\tau_t^{C\to v}$ the allocation of a coalition C to task v at a certain time t. We define the value of an allocation of a coalition C to a task v at time t as $u(\tau_t^{C\to v})$.

In the centralized case, where all the coalitional values are the same to all agents, and agents are aimed to build coalitions that maximize a global utility, this u is a scalar. In the decentralized version of the problem, where coalitional values are different for different agents and each agent tries to maximize its own utility, u is a vector, whose elements u_a are the particular values of coalition C to a task v at time t seen by agent a. In this paper, we focus on the latter version of the problem. Lets define $\Phi' \subseteq \Phi$ as any set of time-feasible agent allocations,[15] and $\boldsymbol{\Gamma}(\boldsymbol{\Phi}', t_{\max}) = \{\tau_t^{C\to v} \mid C = \{a \mid \tau_t^{a\to v} \in \Phi'\}, v \in \boldsymbol{V}, t \le t_{\max}\}$ as the corresponding set of time-feasible coalition allocations to the set of tasks $\boldsymbol{V}$ over a given time period $0 < t < t_{\max}$. In other words, coalition C is formed if a specific set of agents $a \in C$ contribute to doing task v at time $0 < t < t_{\max}$, this coalition is unique for a particular v and t, and coalitions happening at the same t do not overlap in agent members.

The objective of the coalition formation problem is to find $\Phi' \subseteq \Phi$ and consequently, $\boldsymbol{\Gamma}(\boldsymbol{\Phi}', t_{\max})$ such that:

$$\max_{\Phi'} \sum_{v\in V} u\left(\tau_t^{C\to v}\right), \qquad \tau_t^{C\to v} \in \boldsymbol{\Gamma}(\boldsymbol{\Phi}', t_{\max}). \tag{1}$$

The coalition formation problem is computationally expensive, even under quite restrictive assumptions, due to its NP-completeness.[16]

The first algorithm in the literature that was designed specifically for solving the coalition formation problem among cooperative agents using a distributed strategy was developed by Shehory and Kraus.[17] Other approaches (mostly based on Shehory and Kraus[17]) include negotiation algorithms,[18] anytime dynamic programming algorithms,[19–21] integer programming algorithms,[22] and metaheuristic approaches.[23] Coalition-based models have been applied to

multirobot task allocation. In Ref. 24, modifications of the coalition formation algorithm presented in Ref. 17 are provided for applying coalition theory to the multirobot domain, and results from experiments with real robots are reported. In Ref. 25, polynomial time suboptimal algorithms are proposed for real-time task allocation of Unmanned Aerial Vehicles (UAVs).

According to the taxonomy presented in Ref. 26, the coalitional task allocation problem is equivalent to the XD [ST-MR-TA] problem (cross-schedule dependent [XD] single task [ST] multirobot [MR] time-extended task allocation [TA]). A problem is cross-schedule dependent if the utility of an agent toward a task depends not only on their own schedule, but also on the schedules of other agents that might be potential coalition partners. Single task indicates a problem in which agents perform a single task at a time. Time-extended task allocation refers to problems in which agents plan for multiple time steps ahead. Relevant work concerning the XD [SR-MT-TA] problem includes Koes et al.,[27] which uses a centralized mixed-integer linear programming (MILP) strategy for solving the constrained team task allocation for search and rescue missions, and Korsah,[28] which addresses the single-task multiple-robot time-extended task allocation using a centralized MILP for a wider range of dependencies (e.g., intertask precedence, synchronization of starting times, nonoverlapping constraints, intertask proximity and location constraints, penalties for waiting agents).

Market-based approaches have been demonstrated to be successful for multirobot task allocation. Stentz and Dias introduced the idea of a free market scheme for multiple internal coordination of robots[29] using a decentralized bidding mechanism and a centralized gathering of bids (auctioneer). Later work on solving the XD [SR-MT-TA] problem using a decentralized market-based bidding process with more than one auctioneer (leader robots) includes Guerrero and Oliver[30] and Lin and Zheng.[31]

Several efforts have been carried out on multisatellite task allocation. Augenstein et al.[32] use two MILP strategies for allocating image tasks, data downlinks, and health and safety monitoring in an EOS constellation, using constraints related to hardware (e.g., satellite and ground station single operations at a time, limited pointing agility) and operations (e.g., guaranteeing satellite contact frequency and short convergence period for replanning purposes). Bianchessi et al.[33] propose a heuristic based on tabu search for the problem of selecting and scheduling the observation requests to be satisfied, under operational constraints. In Ref. 34, a protocol for planning cooperative-demanding tasks under communication constraints in satellites is proposed. Several sets of protocols are suggested to avoid conflictive coalition assignments originated due to the restrictions in the network topology between agents. However, the aforementioned paper disregards the optimization and time allocation of tasks as well as the management of the resources needed for doing these tasks.

Near-optimal scheduling can be challenging in terms of computational complexity in run time. Examples from the literature include Wolfe and Sorensen[35] and Wang et al.[36] Relevant work in distributed scheduling for satellite constellations includes Das et al.[37] and Van Der Horst and Noble.[38] In particular, Das et al.[37] try to reduce exchange of information between agents by creating feasible schedules in the first iteration of the negotiation process while losing potential of optimality in assignments. Van Der Horst and Noble[38] use multiple local auctions led by an *auctioneer* agent, which tries to reduce communication efforts by allocating tasks in a single bidding iteration.

Decentralized auction-based greedy algorithms are interesting for satisfying the opportunistic resource allocation requirements of the Federated Satellite System concept. A well-known decentralized auction-based algorithm is the Coupled-Constraint Consensus-Based Bundle Algorithm (CCBBA).[39] The CCBBA is an upgrade of the Consensus-Based Bundle Algorithm (CBBA), developed by Choi et al.[40] for task allocation of UAVs. CBBA can be thought as an algorithm that solves a multiobjective optimization problem, where the objective functions are the utilities that each agent sees in performing certain bundle of tasks. CBBA has been recently proven to converge to a pure strategy Nash equilibrium[41] but not Pareto optimal. CBBA has been also proven to give suboptimal solutions for the single-robot, single-task problem ([SR-ST-TA] in Ref. 26) with at least a 50% of optimality with respect to the optimal objective value of the [SR-ST-TA] problem, given a nonnegative scoring scheme that fulfills a convergence property called Diminishing Marginal Gain (DMG). CBBA's optimality is usually close to 90% and the algorithm has been proven to guarantee conflict-free assignments despite marginal inconsistencies in situational awareness. These inconsistencies could be differences in agents' beliefs about the bid placement times, but nothing that compromises conflict-free allocation of tasks to agents. Its scalability with increasing number of tasks and agents makes it particularly attractive for real-time dynamic environments, since the bidding algorithm runs in polynomial time. The original CBBA algorithm is organized in two phases. A brief description of the CBBA is shown below.

1.2.1 | First phase of CBBA

In this phase, each agent adds tasks to its bundle iteratively. Each agent chooses tasks to add based on maximizing its own internal utility function. Given an initial bundle, $\mathbf{b_a}$, agent a adds the task that will result in the largest marginal increase in utility, and repeats the process until the bundle is full or no more tasks increase the overall utility function of the agent.

Every time an agent a thinks it can win a task j (if and only if the agent can outbid the current highest bidder of that task), it adds j provisionally to its bundle, it places a bid c_a^j into a winning bids list $\mathbf{y_a}$, $(y_a^j = c_a^j)$, stores the iteration number in which the agent has placed its bid in a stamp vector $\mathbf{s_a}$ and enters a self-print in a winner array $\mathbf{z_a}$, $(z_a^j = a)$. This winner array contains the identities of the agents that agent a believes are the winners of the tasks.

1.2.2 | Second phase of CBBA

After the first phase, agents broadcast information about their bids in order to resolve conflicting assignments within the team. Agents, in turn, receive the information broadcast by other mates, and compare if they have been outbid for any task in their bundle. As the bid placed for a task in the first phase usually depends upon the bids of other tasks existing in the agent's bundle, up to that point, if an agent is outbid for a task, it must release it and all subsequent tasks from its bundle. For each message that agent a receives from other agents, a set of rules are

applied in order to decide which action is executed by agent *a* to update its information vectors. These rules are the basis of consensus (i.e., lead the agents to reach conflict-free assignments) and involve comparing the winners array ($\mathbf{z_a}$), the winning bids list $\mathbf{y_a}$, and the stamp vector $\mathbf{s_a}$. The rules can be checked in Ref. 40.

An extended version of the CBBA called ACBBA (asynchronous CBBA) used this time stamp vector for allowing asynchronous handling of messages from other agents.[42] The extension of the CCBBA algorithm described in this paper includes the aforementioned improvements in the rules of the second phase of the algorithm (i.e., the ones that originate the consensus on task winners).

Apart from Ref. 39, several other extensions of the CBBA algorithm have been published lately. For example, in Ref. 43, the CBBA algorithm is modified for handling multiple assignments of tasks to agents, thus allowing to form coalitions. However, it is not clear how the algorithm applies in scenarios with superadditivity, and the algorithm is not capable of giving solutions that include partially completed tasks. In Ref. 44, the CBBA algorithm is extended for handling multiple UAV team assignments. In a first phase, a first loop of the CBBA assigns a particular set of tasks to a specific team. Second, this team replans their previously assigned tasks. In Ref. 44, the teams appear to be predefined and, as in Ref. 43, superadditivity effects are not studied. Superadditive score structures assign scores to partially executed tasks. In our case, this score structure depends on the sensors required by the task and the subset of those sensors actually assigned to agents. The inclusion of this score structure is handled in allocation algorithms using task assignment constraints. For example, let us assume a task *v* is divisible in three different subtasks, namely $v = \{A, B, C\}$. The score awarded to the realization of *v* will depend on how many subtasks of *A*, *B*, *C* will be allocated to the agents. A score $S_v = x$ awarded to the realization of only the parts *A* and *B* is codified using the following constraints: $S_v = x$ *if and only if Part A is executed AND Part B is executed AND Part C is NOT executed*. Simple additive scenarios do not need these constraints, as the score scheme is simply determined by the sum of the scores awarded for the realization of each part.

As the number of sensors that a task requires increases, the number of partial allocation possibilities grows exponentially. Hence, superadditive score structures pose a challenging computational problem especially in task-allocation algorithms, where fast conflict-free allocations depend on the quantity of these assignment constraints.

1.3 | Overview and contributions

In this paper, the modification of the CBBA algorithm proposed by Whitten et al.[39] for allocating tasks to agents under complex coupled constraints (CCBBA) is adapted for planning purposes. In Ref. 39, the CBBA algorithm is modified for allowing time and dependency constraints, thus making possible the introduction of superadditivity to the scenario. Whitten et al.[39] proposes the specific constraints needed for task allocation in superadditive environments for a case of two dependent tasks.

The contributions of this paper with respect to the literature are as follows:

- We propose a utility function that includes costs of displacement and behavioral costs, which are the efforts due to splitting and merging a coalition of agents. Behavioral costs were introduced in distributed auction-based algorithms by Butler and Hays[45] for two-robot coalitions. In this paper, we extend the coalition behavioral cost for handling merging and splitting costs in multiagent coalitions of agents.
- We modify the CCBBA algorithm for handling effective coalition formation in superadditive environments. The constraints needed for task allocation in superadditive environments are extended from the ones presented by Whitten[46] for tasks needing more than two sensors, and several modifications are made to the algorithm to handle these constraints more efficiently and adapting them to our framework. In particular, our algorithm ensures that the proposed coalitions are minimal (i.e., superfluous resources are never spent on doing any tasks).
- We propose an experimental simulation framework based on the *Turtlekit* library,[47] an open source Java-based multiagent simulation platform. In this framework, agents perform sets of tasks using a decentralized planning strategy. Agent's behavior (i.e., states such as *planning state*, *going to task*, *performing task*) is implemented using a state machine. The reader can find more information about the software architecture in Appendix A.

  Tasks and their properties are distributed across the scenario using a factorial design structure.[48] This structure allows the researcher to study the effects that different task/agent property factors have on performance metrics. Such factors can take different discrete possible levels (e.g., high/medium/low concentration of the tasks, high/low resource cost of one step movement per agent), so that experimental units take on some combinations of these levels across all such factors. Subsequently, this framework allows the study of the influence of interactions between factors on the metrics.

The remainder of the paper is organized as follows: in Section 2, the methodology behind this framework is described in detail. The utility function for an individual agent and the modification of the CCBBA algorithm for coalition task allocation are described in this section. In Section 3, a first case study is presented analyzing the effect of time and dependency constraints. In Section 4, a second case study is presented focusing on the effects of component failures. Finally, Section 6 summarizes the contributions of this paper, discusses the limitations of the methodology, and outlines some opportunities for future work.

2 | METHODOLOGY

2.1 | Formulation as a planning problem

Let $\boldsymbol{A} = \{a_1, a_2, a_3, \ldots, a_n\}$ be a set of *n* agents. Each agent *a* is equipped with a set of sensors $\epsilon_{\mathbf{a}} \subseteq \epsilon$, where ϵ is the set of all sensors that exist in a scenario. Generally speaking, this model can handle multiple numbers of different sensors, but we have restricted the application to three for simplicity. These are: $\epsilon = \{IR, MW, VIS\}$, where *IR* stands for infrared,

MW for microwave, and *VIS* for visible. These agents are available to perform a set $\boldsymbol{V}$ of m observational tasks $\boldsymbol{V} = \{v_1, v_2, v_3, \ldots, v_m\}$.

Each task has been modeled as a tuple of three elements $v = \langle \mathbf{x}, \epsilon^v, \boldsymbol{TC} \rangle$, where $\mathbf{x}$ are the spatial coordinates of the task, $\epsilon^v = \{\epsilon_1^v, \epsilon_2^v, \ldots, \epsilon_l^v\}$, $\boldsymbol{\epsilon^v} \subseteq \boldsymbol{\epsilon}$ is the set of sensors needed to complete the task, $l = |\boldsymbol{\epsilon^v}| \leq 3$ is the number of sensors needed to complete the task (e.g., a task v with $l = 2$ would require two sensors to be completed, e.g., "infrared" and "microwave"); $\boldsymbol{TC}$ is the set of time constraints that are particular to the task, $\boldsymbol{TC} = \{t_{\text{start}}, t_{\text{end}}, d, t_{\text{corr}}, \lambda\}$; t_{start} and t_{end} are the earliest and latest time an agent can start the execution of a task, respectively; d is the duration of the task; t_{corr} (decorrelation time) is the maximum time difference between the start times of the subtasks composing a task (e.g., the IR, MW, or VIS part of a task); and λ is an urgency factor, which decreases the score by penalizing late execution of the task. In the studies presented later in this paper, we have set all the time parameters $t_{\text{start}} = 0, t_{\text{end}} = \infty, d = 0$ for simplicity.

Each task v can be allocated totally or partially, depending on the type of sensors of the agents assigned to v. Thus, all the possible partial sensor allocations must be accounted for. That is accomplished by dividing each task v in a set of subtasks $\boldsymbol{J_v}$ consisting of each individual part of a task that can be allocated to a sensor, and taking into account all the ways in which task v can be done partially. $N_{\text{sub}_v} = |\boldsymbol{J_v}|$ is the total number of subtasks per task.

Each subtask j represents taking an image for task v with a specific sensor at a certain level of partiality $K(j) \leq l$. This level of partiality will account for the number of sensors that are eventually allocated to task v. For example, a task v that needs two sensors, namely $\boldsymbol{\epsilon^v} = \{IR, MW\}$, will be subdivided in four different subtasks. The first two, $\{IR\}_{\{MW\}}$ and $\{MW\}_{\{IR\}}$ will represent doing the task completely. The subtask in the base position (e.g., $\{IR\}$ in $\{IR\}_{\{MW\}}$) represents the actual subtask, that is, the one for which the agent places a bid. The subtasks in the subindex (e.g., $\{MW\}$ in $\{IR\}_{\{MW\}}$) represent the set of subtasks that share a mutual dependency constraint with the subtask in the base, that is, subtasks that a coalition mate will perform. Eventually, agents will be able to claim the subtasks in the basis if, and only if, the subtasks in the subindex are allocated to themselves or other agents. Intuitively, $K(\{IR\}_{\{MW\}}) = K(\{MW\}_{\{IR\}}) = 2$. $\{IR\}_{\{\}}$ and $\{MW\}_{\{\}}$ will represent doing only the IR or the MW part of the task (the partiality of the aforementioned tasks is $K(\{IR\}_{\{\}}) = K(\{MW\}_{\{\}}) = 1$). Thus, the total number of subtasks per task N_{sub_v}:

$$N_{\text{sub}_v} = |\boldsymbol{J_v}| = \sum_{i=0}^{l} \binom{l}{i} i. \tag{2}$$

For $l = 3$, there are a total of $\binom{3}{1} \cdot 1 + \binom{3}{2} \cdot 2 + \binom{3}{3} \cdot 3 = 3 + 6 + 3 = 12$ subtasks. The reader can find all the subtask allocation possibilities for a task v containing three sensors (e.g., "infrared," "microwave," and "visible"), $\epsilon^v = \{IR, MW, VIS\}, l = 3$ in Table 1.

This number of subtasks grows worse than exponentially. Luckily, the number of subtasks can be reduced by applying some constraints .

For example, subtask $\{IR\}_{\{MW\}}$ cannot be allocated unless its counterpart subtask $\{MW\}_{\{IR\}}$ is allocated as well. Moreover, subtask $\{IR\}_{\{MW\}}$ cannot be allocated if the subtask representing partial realization of the task, $\{IR\}_{\{\}}$, is also allocated. These constraints defining dependencies between inter-subtask, together with other constraints

TABLE 1 Subtask allocation possibilities for a task v requiring three sensors

Allocation possibilities for $l = 3$		
$K(j) = 3$	$K(j) = 2$	$K(j) = 1$
	$[\{IR\}_{\{MW\}}, \{MW\}_{\{IR\}}]$	$[\{IR\}_{\{\}}]$
$[\{IR\}_{\{MR,VIS\}}, \{MW\}_{\{IR,VIS\}}, \{VIS\}_{\{IR,MW\}}]$	$[\{IR\}_{\{VIS\}}, \{VIS\}_{\{VIS\}}]$	$[\{VIS\}_{\{\}}]$
	$[\{VIS\}_{\{MW\}}, \{MW\}_{\{VIS\}}]$	$[\{MW\}_{\{\}}]$

capturing synchronization aspects, are defined below (see Section 2.2). The constraints are formulated using the matrix-based approach by Korsah,[28] where the cross-schedule dependencies are transformed into an instance of the XD [ST-SR-TA] problem, thus reducing the multi-agent task allocation problem to a single-agent allocation problem with coupling constraints. Specifically, $\mathbf{D}$ is the matrix describing the coupling constraints existing among subtasks in $\boldsymbol{J_v}$. Similarly, $\mathbf{T}$ describes the temporal constraints among subtasks in $\boldsymbol{J_v}$. These matrices are explained in more detail in Section 2.2.

Each subtask is worth a particular score. The score function S is given by the following expression:

$$S = \begin{cases} \frac{S_{\max}}{K(j)} e^{-\lambda(t_a^v - t_{\text{start}})} \alpha\left(\frac{K(j)}{l}\right), & \text{if } t_a^v \in [t_{\text{start}}, t_{\text{end}}] \\ & \textit{and } \epsilon_a \text{ contains the sensor of subtask } j \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where $S_{\max}$ is the score awarded to the complete task if it were done at time t_{start} (i.e., with no time discount), t_a^v is the time at which the agent a arrives to the task v location to complete the assigned subtask, and $\alpha(\frac{K(j)}{l})$ is a monotonously increasing function penalizing tasks performed with degraded conditions (i.e., with fewer sensors than required, $K(j) \leq l$), $\alpha(1) = 1$, $\alpha(\frac{K(j)}{l}) \leq 1$. As mentioned previously, the α function has to verify the following condition to resemble a superadditive environment:

$$\alpha\left(\frac{K(j)}{l}\right) > K(j)\alpha\left(\frac{1}{l}\right). \tag{4}$$

Each agent has a utility function $u_a(j)$ that models its preferences concerning subtasks. Let us start by defining agent a's utility of adding a subtask j to a particular subtask path π_a, $u_a^{\pi_a}(j)$:

$$u_a^{\pi_a}(j) = \max_{\mathbf{x}_a^j, t_a^j} \frac{S}{1 + e^{\gamma\left(|\mathbf{x}_\mathbf{a}^\mathbf{j} - \mathbf{x}^\mathbf{j}|\right)}} - g(\mathbf{X}(\pi_\mathbf{a})) - p(\mathbf{A^v}, \mathbf{A^{v-1}}) - c_v, \tag{5}$$

where $\boldsymbol{x_a^j}$ and t_a^j are the spatial and temporal coordinates where agent a decides to perform subtask j; $\boldsymbol{\pi_a}$ is the agent's path (a sequence of tasks); γ is a parameter that dictates the shape of this sigmoid function; $|\mathbf{x}_\mathbf{a}^\mathbf{j} - \mathbf{x}^\mathbf{j}|$ gives the distance between the coordinates where agent a decides to perform j, $\boldsymbol{x_a^j}$, and the physical location of j, $\boldsymbol{x^j} = \boldsymbol{x^v}$; and $\mathbf{X}(\pi_\mathbf{a})$ contains the spatial coordinates vector for each stop in the path.

Note that $g(\mathbf{X}(\pi_\mathbf{a}))$ is the cost that agent a spends on traveling through all the subtask locations in the path. This cost function is linear with respect to the path length, that is $g = \eta \cdot \textit{Total Path Length}$, where

η is the cost of displacement per unit of path length. Note that the path length could be different from $|\mathbf{x}_{\mathbf{a}}^{\mathbf{j}} - \mathbf{x}^{\mathbf{j}}|$, although in all cases presented in the paper the two quantities are equal.

The combined effect of $(\frac{S}{1+e^{\gamma|\mathbf{x}_{\mathbf{a}}^{\mathbf{j}} - \mathbf{x}^{\mathbf{j}}|}} - g)$ is a function in which utility increases as the agent gets closer to the task until a maximum is reached, and then decreases slowly until the distance of the agent to the task is 0. This cost function avoids excessive costs in traveling to the specific coordinates of the task, as in most cases the agents are capable of taking adequate images from a distance. The sigmoid function would be close to 1 within a particular radius of the task and will drop to 0 quickly beyond this radius. When agents calculate the physical coordinates where they are going to take the picture, they will most likely calculate the point at this maximum, as the score of the sigmoid will be almost the same as taking the picture at the exact coordinates of the task, but the agent will save the cost of displacement from that radius to the exact location of the task. Note that this function could be replaced by any other function that is more representative of a particular scenario; however, it retains the basic dependency of score with distance. Thus, $\boldsymbol{x}_{\boldsymbol{a}}^{\boldsymbol{j}}$ can be adjusted so as to maximize the overall utility that each agent sees for a particular task. However, for simplicity, all case studies presented here assume $\gamma \to -\infty$, which results in the agents always performing the tasks exactly at the task location, $\boldsymbol{x}_{\boldsymbol{a}}^{\boldsymbol{j}} = \boldsymbol{x}^{\boldsymbol{v}}$, $\forall a, j, v$.

Note that $p(\mathbf{A}^{\mathbf{v}}, \mathbf{A}^{\mathbf{v-1}})$ is the penalty cost associated with the operations of merging or splitting a coalition. Coalition formation costs were introduced initially by Butler and Hays[45] in auction-based task allocation algorithms. This additional cost may relate to the costs of the exchange of resources and information among the members of the coalition, such as the cost of performing a docking/undocking maneuver. In this paper, we model this cost as a function of the identity of the agents allocated to the task (or partially allocated, if the task is done with degraded conditions), $\boldsymbol{A}^{\boldsymbol{v}} \subseteq \boldsymbol{A}$, and those allocated to the previous task in the path, $\boldsymbol{A}^{\boldsymbol{v-1}} \subseteq \boldsymbol{A}$.

For example, let us consider agent 1, who carries an "infrared" sensor ($\epsilon_{\mathbf{1}} = \{IR\}$), is calculating its utility toward performing subtask j, which belongs to task v. Task v is doable with two sensors, namely "infrared" and "microwave," this is $\epsilon^{\mathbf{v}} = \{IR, MW\}$. Let us assume the "microwave" subtask of task v has already been claimed by agent 2, and therefore $\boldsymbol{A}^{\boldsymbol{v}} = \{1, 2\}$. Let us further consider that agent 1 wants to calculate the penalty toward going to task v after task $v-1$, that is, $t_1^v - t_1^{v-1} > 0$. Let us assume task $v-1$ is also doable with two sensors, $\epsilon^{\mathbf{v-1}} = \{IR, MW\}$, but its coalition mate is another agent 3, different from agent 2, that is, $\boldsymbol{A}^{\boldsymbol{v-1}} = \{1, 3\}$. In that case, agent 1 will have to comply with the cost associated with both splitting its former coalition in $v-1$ with agent 3 and merging with its new coalition mate agent 2. Therefore, $p = C_{\text{split}} + C_{\text{merge}}$, where C_{split} and C_{merge} are the costs of splitting and merging a coalition per agent and per subtask.

If its coalition mate in $v-1$ had been 2 instead of 3, then $\boldsymbol{A}^{\boldsymbol{v-1}}$ would have been $\boldsymbol{A}^{\boldsymbol{v-1}} = \{1, 2\}$, and $p = 0$, as no extra effort would be needed to break an existing (or create a new) coalition.

Note that c_v is the "intrinsic" cost associated with performing the task (e.g., the energy used to take an image). The intrinsic cost depends on the task in principle, although for simplicity it is the same for all case studies presented in this paper.

Since the utility an agent gets from doing a subtask depends on the path, agents will choose the path (sequence of subtasks) that maximizes their individual utility:

$$u_a(j) = \max_{\boldsymbol{\pi_a}} u_a^{\boldsymbol{\pi_a}}(j). \tag{6}$$

In the first phase of the algorithm, agent a will add the subtask that maximizes $u_a(j)$ to the bundle $\boldsymbol{b_a}$, together with the correspondent optimized path $\boldsymbol{\pi_a}$ (see 1).

Algorithm 1. First Phase of the Modified CCBBA for an agent a

procedure FIRSTPHASE(ζ)
  if $\zeta = 0$ **then** ▷ ζ is the iteration number
    INITIALIZEVARIABLESTO0
  else
    $\mathbf{y_a}(\zeta) = \mathbf{y_a}(\zeta - 1)$ ▷ Winner bids list
    $\mathbf{z_a}(\zeta) = \mathbf{z_a}(\zeta - 1)$ ▷ Winner agents array
    $\mathbf{tz_a}(\zeta) = \mathbf{tz_a}(\zeta - 1)$ ▷ Arrival times array
    $\mathbf{c_a}(\zeta) = \mathbf{c_a}(\zeta - 1)$ ▷ Self bids list
    $\mathbf{s_a}(\zeta) = \mathbf{s_a}(\zeta - 1)$ ▷ Iteration stamp vector
  end if
  $\mathbf{v_a^{avail}} \leftarrow$ GETAVAILABLESUBTASKS()
  while number of different tasks in $\mathbf{b_a} < M \& \exists\, h_a^j = 1$ **do**
    for all j **do**
      $\Pi_a \leftarrow$ GETPOSSIBLEPATHS($\mathbf{b_a}, j$) ▷ Possible paths originated when adding j to the bundle
      $c_a^j, t_a^j, x_a^j, i_{opt} \leftarrow$ CALCULATEBIDFORSUBTASK($\Pi_a, j, \boldsymbol{\Omega_a}, \mathbf{tz_a}$) ▷ i_{opt} is the optimum position in the path
      $h_a^j \leftarrow$ COALITIONTEST($c_a^j, j, \mathbf{z_a}, \mathbf{y_a}$)
      if $h_a^j = 1$ **then**
        $h_a^j \leftarrow$ MUTEXTEST($c_a^j, j, \mathbf{z_a}, \mathbf{y_a}$)
      end if
    end for
    $j_{chosen} = \text{argmax}_j\, c_a^j h_a^j$
    $\mathbf{b_a} = \mathbf{b_a} \oplus_{end} j_{chosen}$ ▷ $\oplus_{\bullet}$ indicates addition at position $\{\bullet\}$ of a certain vector
    $\pi_{\mathbf{a}} = \pi_{\mathbf{a}} \oplus_{i_{opt}} j_{chosen}$
    $\mathbf{X}(\pi_{\mathbf{a}}) = \mathbf{X}(\pi_{\mathbf{a}}) \oplus_{i_{opt}} x_a^{j_{chosen}}$
    $\mathbf{t_a} = \mathbf{t_a} \oplus_{i_{opt}} t_a^{j_{chosen}}$ ▷ Vector of execution times
    $y_a^{j_{chosen}} = c_a^{j_{chosen}}$
    $z_a^{j_{chosen}} = a$
    $s_a^{j_{chosen}} = \zeta$
    $tz_a^{j_{chosen}} = t_a^{j_{chosen}}$
    $\boldsymbol{\Omega}_{\mathbf{a}}^{*} \leftarrow$ UPDATECOALITIONMATES($\boldsymbol{\Omega_a}$)
  end while
end procedure

For a simple overview of how the algorithm works, the reader can find in Appendix B a worked out example.

Calculating the optimal path can be very expensive ($O(m!)$) and usually, only the subset of paths that keep the order of previously assigned tasks are considered[49] (i.e., the bidding strategy is greedy to some extent).

The purpose of the decentralized planning algorithm is to make assignments of agents to tasks so that each agent tries to maximize its own utility. Conceptually, the problem that is being solved in this paper can now be formulated as:

$$\max_{\Delta_a^v} \left(\sum_{v \in V} \sum_{j \in J_v} u_a(j) \Delta_a^v \right) \quad \text{for each } a \in \boldsymbol{A}. \tag{7}$$

Subject to the following constraints:

$$\sum_{v \in V} \Delta_a^v \leq M \qquad \forall\, a \in \boldsymbol{A}, \tag{8}$$

$$\sum_{a \in A} \Delta_a^v \leq l \qquad \forall\, v \in \boldsymbol{V}, \tag{9}$$

where Δ_a^v is a binary variable that can be derived from the agents' paths and is equal to 1 if any subtask from v is assigned to a, and 0 otherwise, $\Delta_a^v \in \{0, 1\}\ \forall (a, v) \in \boldsymbol{A} \times \boldsymbol{V}$. M accounts for the maximum number of tasks that an agent can bid for (i.e., the planning horizon). The higher M is, the more computational effort is needed for the algorithm to reach convergence.

While our framework uses a decentralized formulation as defined in this section, a centralized formulation is also defined for benchmarking purposes that maximizes the sum of the agents' utilities. The interested reader can find the centralized formulation of this optimization problem in Appendix C. This centralized formulation has been extended from the MILP formulation of the XD [ST-MR-TA] problem in Korsah.[28]

In this paper, the problem formulated above is solved using a new modified CCBBA algorithm. CCBBA[39] is an existing extension of the CBBA algorithm[40] that includes the possibility of allocating tasks with coupled assignment constraints. We proceed to describe the modifications done to the CCBBA algorithm.

2.2 | Extension of CCBBA to include coalitions

In this paper, we modify the CCBBA algorithm so that agents can bid for any of the subtasks belonging to one particular degree of partiality $K(j)$ (see Table 1) and thus aim for the formation of coalitions. The CCBBA adds the possibility of including coupled constraints in the allocation of these subtasks. The coupled constraints describe any situation where the decisions regarding one agent alter the options available to another agent.

We use these coupled constraints to extend the superadditive score awarding system proposed by Whitten[46] for handling tasks requiring more than two sensors.

Two main types of coupled constraints described in Ref. 46 are considered in this problem:

Mutual dependency constraints: A subtask sharing a mutual dependency constraint with another subtask cannot be allocated unless the counterpart is allocated. All subtasks with a level of partiality $K(j) > 1$ have mutual dependency constraints. For example, if an agent carrying an "infrared" sensor aspires to perform the "infrared" subtask of a task requiring an "infrared" and a "microwave" sensor, the agent would not be able to do that, unless it finds a coalition mate with a "microwave" sensor that also wants to add the task to its bundle.

Mutual exclusive constraints (mutex): A subtask sharing a mutual exclusive constraint with another subtask cannot be allocated if the counterpart is allocated. Subtasks with different $K(j)$ are mutually exclusive. For example, it would be senseless if the agent carrying the "infrared" sensor in the previous example performed both the "infrared" task representing doing the task completely (thus having to find a coalition mate) and partially (without a coalition mate).

In Ref. 39, dependency and mutual exclusive constraints are represented by a *dependency matrix* **D**, a square matrix where rows and columns represent subtasks and each cell (j, q) describes the constraints between subtasks j and q. Dependency constraints are represented by a 1, mutual exclusive constraints are represented by a -1, and a 0 indicates no constraints. An example of **D** for a task with $l = 2$ is shown below. Subtask order of rows and columns follows $j = [\{IR\}_{\{MW\}}, \{MW\}_{\{IR\}}, \{IR\}_{\{\}}, \{MW\}_{\{\}}]$

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}.$$

If there are dependency or mutex constraints, there may also be temporal constraints. In order for the agents to form a coalition that achieves the maximum score of a task, they must arrive to the task location at the same time, or within a maximum delay consistent with an allowed decorrelation time of the particular observation task (see Ref. 50 for typical values of this decorrelation time). Two images taken at a time difference $t_{\text{img1}} - t_{\text{img2}} > t_{\text{corr}}$ will be decorrelated. In other words, if two images taken by the two agents were captured with a relative time delay longer than t_{corr}, the information from the images may no longer be related to the same phenomena, as the observable could have changed substantially.

This information can be encoded in another matrix containing the max decorrelation time allowed for each pair of subtasks. Specifically, let **T** be the matrix that gives the maximum allowed difference in time arrival between two dependent subtasks j and q. Then, by definition:

$$\mathbf{T}(j, q) = \begin{cases} t_{\text{corr}}, & \text{if} \quad \mathbf{D}(j, q) = 1 \\ \infty, & \text{otherwise,} \end{cases} \tag{10}$$

where we assumed the same decorrelation time t_{corr} for all dependent subtasks, although that may not always be the case. Following the same example and subtask order as the **D** matrix, the **T** matrix for a $l = 2$ task would be:

$$\mathbf{T} = \begin{bmatrix} \infty & t_{\text{corr}} & \infty & \infty \\ t_{\text{corr}} & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}.$$

These matrices are tailored to the type of problems we want to solve, but the algorithm can handle multiple types of coupled constraints through different **D** and **T** matrices, which can be useful for future developments of this framework.

Finally, the search for coalition mates is *opportunistic* in the sense that a coalition will be formed only if the agents involved want to allocate the same task at an agreed time slot. In Ref. 39, during the negotiation of the task allocation, an agent that wants to form a coalition to aspire to higher rewards can bid for a subtask a certain number of times (w_{solo}) even if its dependent subtasks are not allocated yet (optimistic strategy). This strategy is also implemented in this paper in order to facilitate the formation of coalitions. If an optimistic strategy is used for bidding on a task and the number of bid attempts (w_{solo}) is

exceeded, the agent can no longer bid on a dependent subtask unless all the dependent subtasks are allocated.

2.3 | Ensuring minimality of agents in the coalition formation process

We extend the CCBBA for ensuring that these coalitions are minimal and no superfluous agents are present in the coalition. For example, a task that requires both an "infrared" and a "microwave" sensor can be addressed in multiple ways. If an agent a_1 carrying both sensors did just the "infrared" subtask and made a coalition with another agent a_2 that carried a "microwave" sensor, the resources spent by agent a_2 would have been wasted since a_1 could have performed the task by itself. This is what we call "minimality of agents" in the coalition formation process.

This minimality in the size of coalitions is enforced during the two phases of the *Modified* CCBBA algorithm. In the first phase, it is carried out via the CoalitionTest subroutine (see Algorithm 1, line 14).

This elimination of waste of resources works by facilitating winning a particular subtask j if the bidding agent has previously won all other dependent subtasks. Specifically, the bid placed for each subtask, c_a^j, is compared to the sum of the bids of the subtasks that share a dependency constraint with j ($\mathbf{D}(j,q)=1$) rather than the bid placed on the single subtask j. This is implemented through an availability indicator h_a^j that dictates whether subtask j is available to agent a:

$$h_a^j = \begin{cases} 1, & \text{if } c_a^j + \sum_{k\in J_v} y_a^k > \sum_{q\in J_v} y_a^q \ \forall\, q,k \text{ such that} \\ & z_a^q = z_a^j \quad \& \quad \mathbf{D}(j,q)\in\{0,1\} \\ & z_a^k = a \quad \& \quad \mathbf{D}(j,k)=1 \\ 0, & \text{otherwise,} \end{cases} \tag{11}$$

where the jth entry of the availability indicator h_a^j is 1 if subtask j is available to agent a, and 0 otherwise. Notice that if $j\in \mathbf{J_v}$, then $k,q\in \mathbf{J_v}$. Notice how the simple CBBA[40] is included in the case $\mathbf{D}(j,q)=0$: when no dependencies of any type are considered between tasks, $\mathbf{J_v}=\{j\}$, that is, $|\mathbf{J_v}|=1$ and $j=q$. Therefore, $\mathbf{D}(j,q)=0$. As no subtask k exists in $\mathbf{J_v}$ such that $\mathbf{D}(j,k)=1$, then $k=\emptyset$ and Equation 11 reduces to $h_a^j = \mathbb{I}(c_a^j > y_a^j)$, where $\mathbb{I}(\bullet)=1$ if condition in $\bullet$ is fulfilled, and 0 otherwise.

In the second phase of the *Modified* CCBBA algorithm, a similar process to the elimination of waste of resources is performed in the first phase is done. Instead of comparing the believed maximum bids for each subtask (y_a^j), the total *participation* of each agent in the task is calculated. *Participation* of agent a on subtask j, f_a^j is defined as the sum of all the bids that an agent believes it has won from all the subtasks $j\in \mathbf{J_v}$.

That is:

$$f_a^j = \sum_{q\in J_v} y_a^q, \ \forall\, q \text{ such that } z_a^q = a \quad \text{and} \quad \mathbf{D}(j,q)\in\{0,1\}. \tag{12}$$

It can be inferred from the previous equation that $f_a^j = f_a^q$ for all j,q that fulfill $z_a^j = z_a^q = a \ \&\ \mathbf{D}(j,q)\in\{0,1\}$.

The importance and effectiveness of this mechanism to ensure minimality of agents is demonstrated in Section 3.

2.4 | Facilitating the convergence of the *Modified* CCBBA

In order to avoid conflict formation in the second phase of our *Modified* CCBBA, and thus foster convergence and increase optimality, agents are only allowed to place bids in one possible combination of subtasks per task. To achieve this, it is necessary to discern which combination of dependent subtasks provides a better assignment solution. For this reason, a test on mutexed combinations of tasks is performed (MutexTest) by each agent in the first phase of the CCBBA algorithm.

This test determines if an agent is allowed to bid on a subtask j and modifies the availability digit h_a^j accordingly. Specifically, the bid is allowed as long as the agent can place a bid larger than the winning sum of the bids of all subtasks mutexed with j. That is:

$$h_a^j = \begin{cases} 1, & \text{if } c_a^j + \sum_{q\in J_v} y_a^q > \max_{J'_v} \sum_{k\in J'_v} y_a^k \\ & \forall\, q\neq j \text{ such that } \mathbf{D}(j,q)=1 \\ & k\neq j \text{ such that } \mathbf{D}(k,j)=\mathbf{D}(k,q)=-1 \\ 0, & \text{otherwise,} \end{cases} \tag{13}$$

where $J'_v \subseteq J_v$ is any set of all the subtasks $j,q\in \mathbf{J_v}$ such that $\mathbf{D}(j,q)=1$.

In the second phase of this *Modified* CCBBA, the same condition is checked to force agents to release their subtasks in case they were outbid by any other group of agents bidding for a mutexed combination of subtasks.

2.5 | Introducing engagement costs to coalition formation

In our *Modified* CCBBA, agents keep track of the coalition mates of tasks that need sensor collaboration. This information is placed in the Coalition Mates Matrix:

$$\mathbf{\Omega} = \begin{bmatrix} \omega_{11} & \omega_{12} & \cdots & \cdots & \omega_{1\text{end}} \\ \omega_{21} & \omega_{22} & \cdots & \cdots & \omega_{2\text{end}} \\ \vdots & \vdots & \omega_{ki} & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \omega_{l1} & \omega_{l2} & \cdots & \cdots & \omega_{l\text{end}} \end{bmatrix}, \tag{14}$$

where rows are agents, columns are tasks, and $\omega_{ki}\in \mathbf{A}$ represents the identity of the kth ($k\leq l$) coalition mate in task i ($i\leq \text{end}, \text{end}\leq M$).

Coalitional intrinsic costs are introduced in our *Modified* CCBBA in the two phases. In the first phase, agents compute the price of merging and splitting a coalition as a function of the available information they have about the environment. This process is included in subroutine CalculateBidForSubTask (Algorithm 1, line 14).

Coalitional intrinsic costs are function of the identities of the coalition mates with whom agent a is going to perform the allocated tasks. Therefore, these costs are defined for tasks, as opposed to the utilities, which were defined before for subtasks. Given a task path $\boldsymbol{\pi}_{\mathbf{a}}^{(\mathbf{v})} = \{v_1, v_2, \ldots, v_i, \ldots, v_{\text{end}}\}$, $\text{end}\leq M$, ($v_i \in \boldsymbol{\pi}_{\mathbf{a}}^{(\mathbf{v})}$ if there exists j such that $j\in \mathbf{J_v}$ and $j\in \boldsymbol{\pi}_{\mathbf{a}}$), each agent a loops over all the tasks included in the path and subtracts the coalition intrinsic costs for task v_i based on the coalition mates of the task prior to the currently evaluated task v_{i-1}. That is, agent a checks the identity of $\omega_{1(i-1)}, \omega_{2(i-1)}, \ldots, \omega_{l(i-1)} \in \mathbf{\Omega_a}$.

The same process is performed for each evaluated task v_i. For each coalition mate, if it accompanies agent a in its prior task but does not do so in the current task, the algorithm considers that there has been a split of the prior coalition, and a split penalizing cost is added to the total engagement costs. Similarly, if a mate accompanies agent a in its current subtask but does not do so in the prior subtask, the algorithm will consider that there has been a formation of a new coalition, and a merge penalizing cost is added to the engagement costs.

During the second phase of the algorithm, our *Modified* CCBBA algorithm checks if the agent's engagement costs are no longer consistent with the allocation situation. This is done by adding a COALITIONSAT flag using a similar process to the MUTEXSAT, DEPSAT, TEMPSAT in the original CCBBA.[46] Each agent constructed its path with its available information about who were its coalition partners. After the consensus process, these coalition partners may have changed or disappeared. If no members of a coalition regarding a task v_i have been found ($\omega_{1i} = \omega_{2i} = \cdots = \omega_{li} = \emptyset$), but there has been an allocation of a winner to a particular subtask $j \in \mathbf{J_v}$ ($z_a^j \neq \emptyset$) that needs a coalition mate, the agent will have to release j to force recalculation of the costs and account for the costs derived by joining a new coalition or splitting the latter one. If $\omega_{ki} \neq \omega_{ki-1} \neq \emptyset$, that is, the kth coalition mate of agent a has changed, and is not $\emptyset$, it will release the task as well for two reasons: first, the temporal constraints of the new mate could be more favorable for agent a (e.g., agent a can arrive at different times t_a^j to the subtasks $j \in \mathbf{J_v}$); and second, the agent may need to update engagement costs as well, as the identities of its mates have changed (i.e., $\mathbf{z_a}$ is no longer consistent with $\mathbf{\Omega}$). $\mathbf{\Omega}$ is not updated in the consensus phase, it is cleared if tasks are released. Coalition mates will be updated in first phase of next iteration, in order to recalculate merging and splitting costs if necessary. For example, in iteration 1, agent a has a coalition mate $\omega_{ki}^{(1)}$, where the subscript 1 indicates iteration 1. The agents become a coalition at task v_{i-1}, thus $\omega_{ki-1}^{(1)} = \omega_{ki}^{(1)}$. In this case, no coalition merge or split costs are needed. In iteration 2, another agent different from $\omega_{ki}^{(1)}$, that is $\omega_{ki}^{(2)}$, wins the subtask and substitutes $\omega_{ki}^{(1)}$ as coalition mate for agent a at task v_{i-1}. In this case, a coalition merge and a split have occurred. Merging and splitting costs need to be recalculated.

2.6 | Optimality of the *Modified* CCBBA

When proposing a new algorithm, it is important to analyze its optimality. The original CBBA proved to have an optimality bound of a 50% of the [SR-ST-TA] problem.[40] Due to the task elimination system (token system),[46] this fundamental guarantee does not apply. Therefore, we rely on simulation to assess the optimality of the *Modified* CCBBA. We are particularly interested in comparing the optimal solutions found with *Modified* CCBBA with those found using a centralized planning strategy.

With this purpose, we also implemented the centralized version of the XD[ST-MR-TA] problem with engagement costs and solved it using mixed integer quadratic programming (MIQP). This formulation can be found in Appendix C and has been extended from Ref. 28. The results of the centralized approach were compared to those of *Modified* CCBBA (decentralized approach) on the same set of test scenarios.

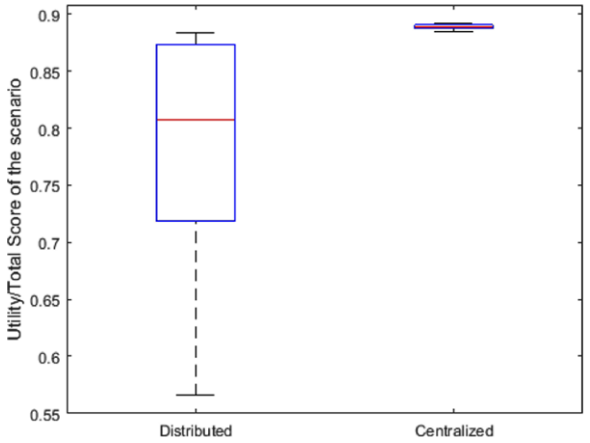

FIGURE 2 Comparison between the utilities obtained by agents following the centralized version of the XD [ST-MR-TA] problem and the agents subscripting to the *Modified* CCBBA algorithm (decentralized approach to the XD [ST-MR-TA] problem)

Twenty-two test scenarios were generated variating the position randomly of three tasks doable with two sensors (e.g., $\{IR, MW\}$), and eight doable with one sensor (e.g., $\{IR\}$ or $\{MW\}$). There were four agents in the simulation, two of them carrying an $\{IR\}$ sensor and two of them carrying a $\{MW\}$ sensor ($\epsilon_{a_1} = \epsilon_{a_2} = \{IR\}, \epsilon_{a_3} = \epsilon_{a_4} = \{IR\}$).

Task intrinsic costs c_v were set to 0.30% of the total resources per agent. The traveling costs were 0.20% of the total resources per agent per step.

Merging and splitting costs C_{merge} and C_{split} were 0.30% and 0.15% of the total resources per agent, respectively. The time discount parameter was $\lambda = 0$.

Results gave a cumulative utility mean of 0.88 for the centralized case, and 0.77 for the decentralized case. Standard deviations were 0.002 and 0.090, respectively. The reader can find comparative boxplots of the results of the study in Figure 2. We can observe how in the cases simulated, the results for the *Modified* CCBBA were not worse than the 50% of optimality proven for CBBA. Note that 90% of the simulations yielded utilities higher than the 80% of the mean reported for the centralized approach.

These results are limited by the relatively small range of scenarios considered in terms of tasks, agents, and sensors. A more extensive optimality study was left out of the scope of this paper due to the costly computations for scenarios with a high number of agents and tasks and the difficulty of including engagement costs using a centralized MIQP approach for agents carrying more than one sensor.

3 | VALIDATION OF THE ALGORITHM

This section describes five studies that contributed to the validation of the algorithm while characterizing the dependence of results on major model parameters, namely the time and dependency constraints, number of decision makers, engagement costs, and planning horizon. A study of the utility of routine for the elimination of waste of resources is also included.

3.1 | Quantifying the influence of the time and dependency constraints

In this section, the extent to which the time and assignment dependency constraints affect the overall performance of the system is analyzed. The task and agent properties used in this case study are described below.

Task properties: Agents perform 30 randomly positioned tasks in a 50 × 50 unit length scenario. These tasks also have a randomly distributed score.

Note that 50% of them require two sensors (e.g., $\{IR, MW\}$) to be fully satisfied. Partially satisfied tasks earn 33% of the total score. The other 50% of the tasks only need one sensor for achieving the maximum score, either $\{IR\}$ or $\{MW\}$.

Agent properties: The path costs ($g(\mathbf{X}(\pi_a))$) are equal to 0.57% of the total resources per unit of distance. No coalition costs are considered in this experiment ($p(\mathbf{A}^{\mathbf{v}}, \mathbf{A}^{\mathbf{v}-\mathbf{1}}) = 0$). Intrinsic costs are equal to 1.14% of the total resources per $\{IR\}$ or $\{MW\}$ image taken. For simplicity, the γ constant decay has been taken as $-\infty$ when each agent decides to take the picture in the place of the task and ∞ otherwise (i.e., the sigmoid is replaced with a step function). This makes the score function to yield 0 unless agent decides to perform subtasks just at the task coordinates. M is the planning horizon.

Two comparable cases have been considered. In the first case ("integral case," or "unconstrained case"), an integral sensor architecture is explored. There are only two agents, both of which carry all sensors required to perform the tasks without the need of forming coalitions, thus $\epsilon_1 = \epsilon_2 = \{IR, MW\}$. Hence, they do not have any dependency or temporal constraints to meet.

In the second case ("modular case," or "constrained case"), a more modular architecture is explored. Four agents carrying one sensor each are considered $\epsilon_1 = \epsilon_2 = \{IR\}; \epsilon_3 = \epsilon_4 = \{MW\}$. In this case, coalitions are needed in order to achieve high scores.

To ensure that the comparison between the two cases is fair, the cost of moving (η) and the cost of doing a task (c_v) in the modular case are half of those in the integral case.

This models the fact that presumably, agents with one sensor would be smaller vehicles and thus require less energy to operate than agents with two sensors. The resources follow the same pattern: the average cost of movement per agent (η) remains the same percentage with respect to the total amount of resources in both cases (0.57% of the total resources per unit of distance). As such, the amount of resources that each agent has in the modular case is half of the amount in the integral case. This models the fact that, for example, those larger vehicles would presumably have larger batteries.

Different values of decorrelation time t_{corr} are considered for the modular case, in order to study the effect of this parameter on performance. Specifically, we measure the number of coalitions formed, the sum of scores for all agents, and the sum of costs for all agents. The number of coalitions formed is normalized by the number of coalitions available to yield a number between 0% and 100%. Similarly, the total score is normalized by the maximum achievable score per scenario, and the total cost is normalized by the total amount of resources per agent.

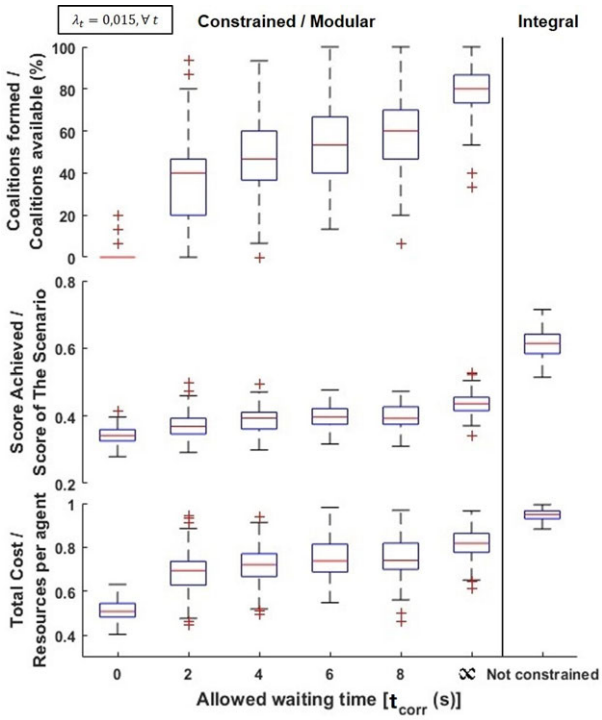

FIGURE 3 Boxplots of the nondimensional score (top), cost (middle), and number of coalitions formed (bottom) for different decorrelation times t_{corr}

Naturally, t_{corr} does not apply in the modular case as no coalitions are formed.

A total of 128 per each t_{corr} scenarios were simulated, with varying task positions and scores.

The results of this study are shown on Figure 3, which shows boxplots for the number of coalitions formed, total score, and total cost, for each value of t_{corr} in the modular case, and a boxplot for total score and total cost for the integral case (in the far right of Figure 3).

In the modular case, coalesced agents need to satisfy meeting time decorrelation constraints to aspire to higher rewards. When t_{corr} increases, mutual *opportunistic* agreement becomes easier, as a higher margin of waiting time is given to the agent for waiting the others for building a coalition. As expected, agents form more coalitions and achieve better scores as t_{corr} increases in the modular case (see Figure 3).

Initially, increasing the allowed time of decorrelation results in higher rewards because the agents have more time for coordinate their schedules (two-sample t-test between data from the score obtained in $t_{corr} = 0$ and $t_{corr} = 2$, $p = 4.1E-13$; $t_{corr} = 2$ and $t_{corr} = 4$, $p = 2.5E-3$; $t_{corr} = 4$ and $t_{corr} = 6$, $p = 7.4E-3$.) After a certain threshold ($t_{corr} = 6$), we observe a somewhat asymptotic behavior, probably explained by the fact that the decorrelation time is large enough to allow for most coalitions to be formed ($t_{corr} = 6$ and $t_{corr} = 8$, $p = .70$). In addition, we observe that the score for infinite decorrelation time in the modular case is significantly lower than the score obtained in the unconstrained case ($t_{corr} = \infty$ and non-constrained case, $p < 8.8E-110$). The t-test has been done using a level of significance of $\alpha = 0.05$ and applying the Bonferroni correction for the number of tests. The tests do not reject

the null hypothesis for this values of α for $t_{corr} \geq 6$ in the constrained case.

Before seeing these results, one could have predicted that the constrained or modular case, even though it has more constraints in the assignments, would result in higher scores, as agents can move freely to where they think is the best target to maximize their utility, whereas in the integral case, sensors cannot be split to address different tasks. However, this is not consistent with the results.

We believe this is due to two factors: (1) The number of maximum tasks an agent can bid for is restricted. Therefore, an agent willing to do a task requiring another sensor might not find any agent that would like to do that task, even if the overall score is maximized by addressing that specific task. (2) The overall score is the sum of the scores of the agents. Recall that agents bid opportunistically and despite the algorithm being collaborative, agents are greedy when placing their bids. In the nonconstrained or integral case, there are two decision makers, whereas in the constrained or modular case, there are four. If decisions are made thinking only on the self-benefit, the solution achieved will degrade as the number of agents increase. This is consistent with the results of the next study.

3.2 | Quantifying the influence of the number of agents/decision makers

In this section, we analyze the influence of the number of agents on the score metric. We simulated three mutually fair cases. In these cases, randomly positioned agents address tasks which are also scored and positioned randomly across the scenario. Values of $t_{corr} = 2s$, $\lambda = 0.17$ are constant across all scenarios. Agent properties with respect to path costs per unit of distance, intrinsic costs, and spatial exponential decay are the same as in Section 3.1.

First Case (1 sensor type): Three agents carry only an IR sensor. Agents address all the tasks without making any coalition (in the previous example, this would be the integral case) Thus, $\epsilon_1 = \epsilon_2 = \epsilon_3 = \{IR\}$. There are 30 tasks that only require an IR sensor, that is, $\epsilon^v = \{IR\}$.

Second Case (2 sensor types): Three agents carry an IR sensor and three agents carry a MW sensor. Resources and travel costs per unit distance are half the ones in the first case. There are 30 tasks doable with two sensors, that is, $\epsilon^v = \{IR, MW\}$.

Third Case (3 sensor types): Three agents carry an IR sensor, three agents carry a MW sensor, three agents carry a VIS sensor. Resources and travel costs per unit distance are one-third of the ones in the first case. There are 30 tasks doable with three sensors, that is, $\epsilon^v = \{IR, MW, VIS\}$.

The aim of this experiment is to analyze to what extent the number of agents affects the overall performance of the system. Due to the *opportunistic* nature of agents (e.g., greediness), multiple agents could originate more competition and less ability to agree upon meeting dependency constraints for access to higher rewards.

Figure 4 shows how the performance of the agents is degraded as the number of agents increases. The degradation of the utility as a function of time and the presence of a maximum decorrelation time increases the difficulty of reaching conflict-free assignments that are valuable to all the agents involved. This coordination cost is shown in

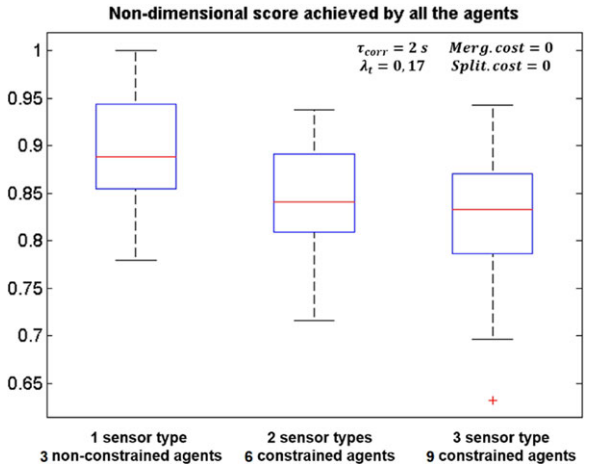

FIGURE 4 Average agent score as a function of number of decision makers

the degradation of the average total score achieved by all the agents. A noticeable point of this analysis is that despite this superadditive scoring scheme in which coalitions are encouraged by offering higher rewards, the *grand* coalition (e.g., a coalition of agents carrying all the sensors needed for the complete allocation of a task),[17] is not always the best solution for addressing such tasks, given the degradation of the utility as a function of the allocation time, the presence of a maximum decorrelation time, and the maximum number of bids per agent (M).

3.3 | Quantifying the influence of engagement costs

In this experiment, only a single scenario is considered. Six single-sensor agents ($\epsilon_1 = \epsilon_2 = \{IR\}; \epsilon_3 = \epsilon_4 = \{MW\}; \epsilon_5 = \epsilon_6 = \{VIS\}$) coordinate themselves to execute 30 different tasks, whose position and maximum awarded score per subtask are distributed randomly. Intrinsic costs, c_v, are equal to 1.14% of the total resources per image taken.

For simplicity, the exponential decay γ has been also been taken as $-\infty$ as in Section 3.1, so agents perform all tasks at the task location. The travel costs per agent are 0.57% of the total resources per unit of distance. Nine tasks require the assistance of three different sensors ($\epsilon^v = \{IR, MW, VIS\}, v \in [1, 9]$); nine of them need two ($\epsilon^v = \{IR, MW\}, v \in [10, 12], \epsilon^v = \{IR, VIS\}, v \in [13, 15]; \epsilon^v = \{MW, VIS\}, v \in [16, 18]$), and 12 of them are doable with only a single sensor ($\epsilon^v = \{IR\}, v \in [19, 22], \epsilon^v = \{VIS\}, v \in [23, 26]; \epsilon^v = \{MW\}, v \in [27, 30]$). We run scenarios for different values of merge costs to measure the effect of this parameter on performance.

Figure 5 shows the evolution of total score and the number of tasks done as a function of merge costs. We observe that, even when the merge cost is 0, agents still show reluctance to form coalitions, due to the presence of dependency and time constraints. Out of nine available tasks doable with three sensors, agents can only do four, and they decide either doing the other five partially, or not doing them at all due to them having negative utility for the agents. As the merging cost increases, agents increasingly refuse to make coalitions. The number of three-sensor tasks done decreases quickly to 0 for merging costs

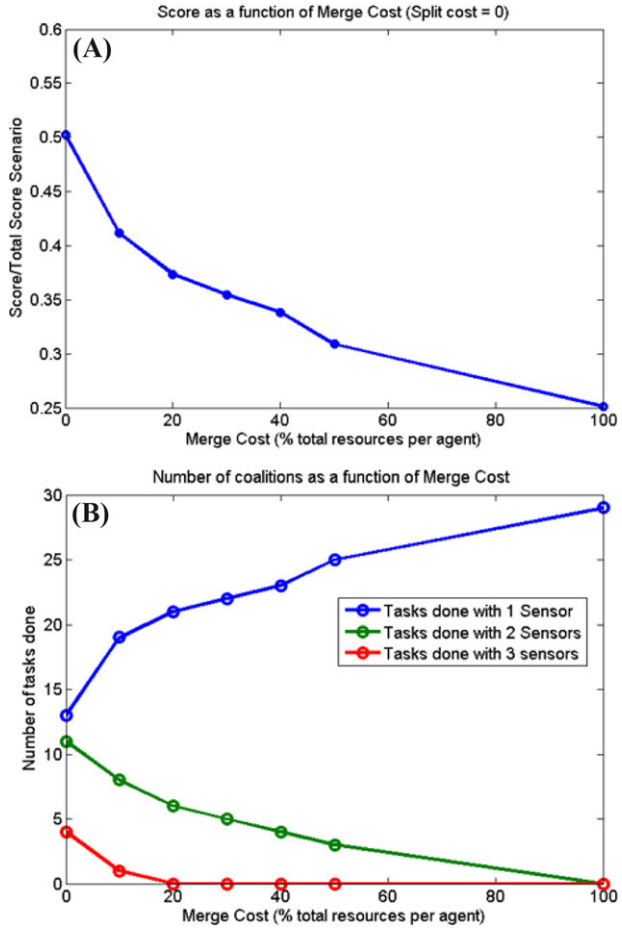

FIGURE 5 Total score (A) and number of tasks done (B) as a function of merging costs

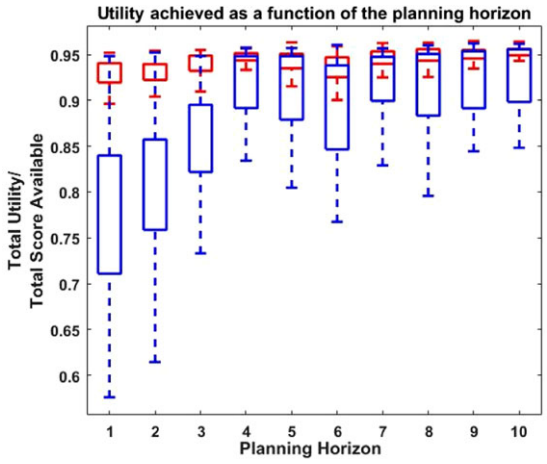

FIGURE 6 Figure shows the utility achieved by the agents under scenarios with different levels of difficulty, easy (red) and hard (blue)

greater than 0. As expected, agents prefer to do a task with degraded score rather than trying to spend resources on aspiring to higher rewards. A similar but less abrupt behavior is observed for two-sensor tasks. Conversely, the number of one-sensor tasks done increases as we increase merge costs. This is due to the fact that agents prioritize one-sensor tasks in the presence of high merge costs. While the exact values observed in Figure 5 depend on the relative values chosen for the scenarios (particularly merge costs as well as task score structure), the overall trends and shape of these curves are as expected.

3.4 | Dependency of performance of the algorithm with the planning horizon

In this experiment, we measured the performance of the agents as a function of the planning horizon *M*.

There are nine agents in this experiment; each one carries one sensor and the type of sensors is balanced such that each of the three groups of three agents carry either an $\{IR\}, \{MW\}$ or $\{VIS\}$ sensor. Agent properties such as path costs, intrinsic cost, total amount of resources, and coalition engagement costs are the same as in the experiment described in Section 3.1 for the constrained case.

We studied the influence of the planning horizon *M* on total score in two different scenarios. The first one is an "easy" scenario, where coalition formation is fueled by the large score dispersion between multisensored tasks. In other words, some tasks are awarded a S_{max}, which is much higher than others. In this case, the bidding process will depend mostly on the scores of traveling to the tasks. Coalitions will be easier to find, as most of the agents will start bidding for the high scored tasks.

The second scenario is a "difficult" scenario, where all the multisensored tasks are worth the same score.

In the second case, the bidding process will depend mostly on the costs of traveling to the tasks. Coalitions will be harder to find, as the random position of the agents will make them start bidding for the tasks that are closer, rather than the tasks with higher score.

Both easy and hard scenarios consist of 30 randomly positioned tasks doable with the same sensor scheme as in Section 3.3.

The results of his experiment are shown in Figure 6, which shows the boxplots of total score for both the "easy" (blue) and "hard" (red) scenarios, for different values of planning horizon. We observe how, after a certain threshold of planning horizon, even in difficult scenarios, the utility achieved by agents cannot improve anymore (Figure 6 A). Beyond this threshold, the algorithm is not able to give better solutions even if it takes into account longer paths. This is because the higher amount of tasks in the bundle also implies a higher probability of conflict in the bidding process.

3.5 | Effectiveness of the elimination of waste resources process to ensure minimality of agents

This experiment validates the effectiveness of the routine for eliminating "sensor waste," or ensuring minimality of agents as described in Section 3.5.

In this simulation, six agents work on a set of 30 different tasks, whose position and maximum awarded score per subtask are randomly distributed.

Agents have two sensors each, these are $\epsilon_{a_1} = \epsilon_{a_2} = \{IR, MW\}$; $\epsilon_{a_3} = \epsilon_{a_4} = \{IR, VIS\}$; $\epsilon_{a_5} = \epsilon_{a_6} = \{VIS, MW\}$. All tasks are doable with two sensors. The identity of these sensors is $\epsilon^v = \{IR, MW\}, v \in [1, 10]$, $\epsilon^v = \{IR, VIS\}, v \in [11, 20]$; $\epsilon^v = \{VIS, MW\}, v \in [21, 30]$). The

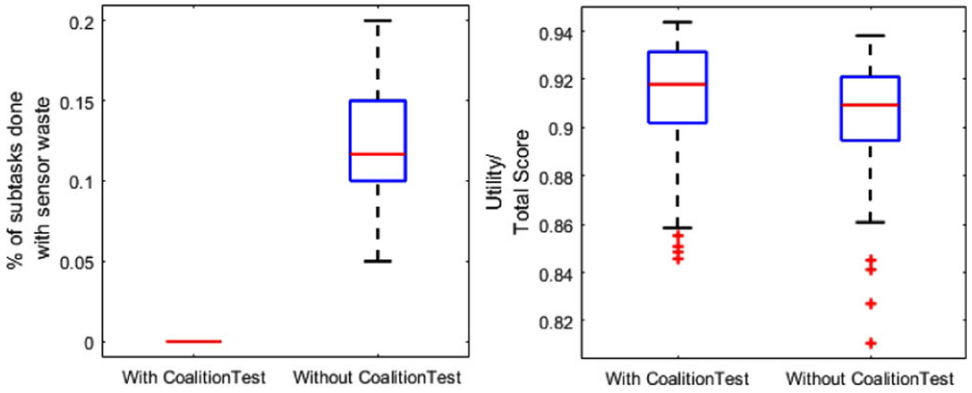

FIGURE 7 Fraction of subtasks done with sensor waste (left) and total score (right) as a function of whether the waste of resources elimination rule is applied

decorrelation time is set to $t_{corr} = 2$. The urgency factor is set to $\lambda = 0.0005$.

The task scoring structure is such that agents are encouraged to bid for the allocation of multisensored tasks. This is done by awarding a higher score per subtask (sensor) to tasks doable with more sensors.

We analyzed the performance of the agents in two different cases. In the first case, the routine of resource waste elimination forces minimality in coalitions. That is, agents carrying the three sensors, will most likely win the allocation of all the subtasks belonging to multisensored tasks. In the second case, the elimination of sensor waste has been suppressed, thus agents will be able to win the allocation of subtasks regardless of the number of sensors they have available. The scoring scheme awards higher rewards to agents that allocate multisensored tasks, thus competition for the allocation of these tasks is higher.

The results of this experiment are shown in Figure 7, which shows the fraction of subtasks done with more sensors than required (labeled "sensor waste," on the left), as well as the total score (right), for both cases (with and without the resource elimination). As expected, in the cases where the elimination of sensor waste routine is applied (Figure 7, left), no subtask is performed with sensor waste. When the routine is not applied (Figure 7, right), we observe that agents participate in tasks and form coalitions unnecessarily, that is, their greediness makes them take some tasks from agents that have enough resources for performing this task on their own. Fully equipped agents lose between 6% and 20% of the subtasks because of agent greediness.

We observe that total scores are also higher when the elimination of waste resources process is active, although the difference is not significant (two- sample *t*-test of the utilities with the elimination of resource waste and without it: *p*-value = .0987, $\alpha = 0.05$).

This indicates that the savings in resources due to this routine do not necessarily translate into score benefits, as scores are measured in terms of utilities of agents.

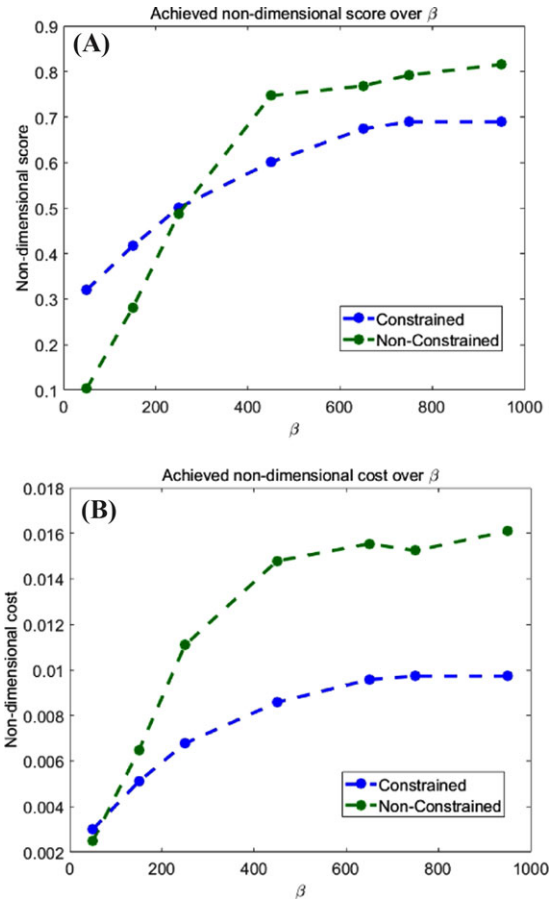

FIGURE 8 Score (A) and cost (B) achieved by the agents for different levels of severity

4 | CASE STUDY: EFFECT OF COMPONENT FAILURES

In this case study, we revisit the question of how the value propositions for monolithic and distributed architectures compare relative to risks resulting from on-orbit failure uncertainties. This question has been analyzed several times in the literature.[5,51] For example, O'Neill[51] did a similar analysis with fractionated spacecraft. They hypothesized that fractionated spacecraft architectures become more valuable in terms of total benefit than monolithic architectures when the system lifetime becomes more severe. The hypothesis was tested using statistical analysis and data from existing missions. In this paper, we use our agent-based simulation approach to study the same question for distributed versus monolithic architectures.

We recall the two different cases proposed in Section 3.1:

a set of homogeneous agents whose sensors are structured in a monolithic architecture (i.e., agents carry enough sensors for doing all the tasks without the need of forming coalitions); and a more distributed architecture, in which the required sensors are distributed across several agents (i.e., agents must form coalitions to achieve higher rewards). Parameters regarding moving costs, intrinsic costs, space dependence of the score, and merging/splitting costs are the same as the ones used in Section 3.1 for the distributed (constrained) and monolithic (nonconstrained) case. Decorrelation time is set to $t_{corr} = 2$. The urgency factor is set to $\lambda = 0.04$. For the monolithic case, there are two agents $\epsilon_1 = \epsilon_2 = \{IR, MW, VIS\}$. For the distributed case, there are six agents $\epsilon_1 = \epsilon_2 = \{IR\}; \epsilon_3 = \epsilon_4 = \{MW\}; \epsilon_5 = \epsilon_6 = \{VIS\}$.

Importantly, we incorporate component failures into the model. In both cases, agents are subject to the failure of any of their sensors or the "bus" (i.e., the platform that provides sensors with the resources they need, such as electrical power, data handling, etc.). For simplicity, the time to failure of components is modeled using an exponential distribution.

The probability density function is shown as

$$P(t^*_{\text{failure}}) = \frac{1}{\beta} e^{\frac{-t^*_{\text{failure}}}{\beta}}, \tag{15}$$

$$t_{\text{failure}_{\text{sensor}}} = \min\{t^*_{\text{failure}_{\text{Sens}}}, t^*_{\text{failure}_{\text{bus}}}\}, \tag{16}$$

where $t^*_{\text{failure}_{\text{Sens}}}$ is the time to failure for the sensor itself without taking the bus into account, and $t^*_{\text{failure}_{\text{bus}}}$ is the time to failure for the bus, both of which follow the probability density distribution in Equation 15.

When an agent discovers the failure of one of its sensors, it inhibits the damaged sensor. If any task is affected by the failure, each agent invalidates its entire current bundle and coalitions and sends a message to the other agents requesting a new plan. Agents stop the execution of their plans and reenter the first phase of the *Modified* CCBBA. Agents with multiple sensors are still allowed to make coalitions with other agents if they lose a sensor.

The results of this study are shown in Figure 8, which show the total score and total cost as a function of the severity (β) for both the monolithic (red) and distributed (blue) case. As shown in Figure 8, in low severity cases ($\beta > 600$), the monolithic case performs much better than the distributed one, due to the known coordination expense. These results agree with the results from Figure 3, in which for highly reliable instruments (no failure was implemented there), monolithic architectures perform substantially better than agent systems forced to collaborate for aspiring to higher rewards. On the other hand, we observe that there exists a value of β below which the unreliability of the components makes distributed architectures more desirable.

In the cost figure, we observe a similar threshold of severity beyond which the costs of the distributed architecture are lower than those of the monolithic architecture. In the distributed case, when a bus fails, only one sensor is affected, since there is one sensor per agent. The other agents live on, and due to the problem formulation, agents with a single sensor move at a smaller cost than multisensored agents. In the monolithic case, a bus failure affects all sensors. For severe environments, the bus usually fails while the agent is moving toward the place from which the task is executed, thus earning 0 rewards and spending resources on the motion. Thus, for high severities, the lower displacement costs are beneficial to the constrained agents (early crossover of the cost metric). When severity is low, the benefits of monolithic architectures against distributed due to coordination costs become predominant.

The effects of severity stagnate when the average time to failure exceeds the total duration of the missions, which explains the observed asymptotic behavior in the score and costs of the agents shown in Figure 8.

5 | DISCUSSION

The framework described in this paper suffers from various limitations. First, as was the case for the CBBA algorithm, our *Modified* CCBBA algorithm is only guaranteed to converge in situations where all the bidding schemes fulfill the DMG condition described in Ref. 40. The DMG is a particular rule in the bidding system that prevents agents from bidding for a particular task in early positions of the bundle than they would for the same task in later positions of the bundle.

All such cases analyzed in this paper have converged to a conflict-free solution in the *Modified* CCBBA algorithm.

Second, the algorithm is relatively expensive computationally—although still less expensive than a centralized algorithm. The worst-case complexity of the *Modified* CCBBA algorithm as a function of the maximum number of sensors per task, $l_{\max}$, number of agents n, number of tasks m, and the planning horizon M is given by the following expression:

$$O\left(\Theta M l_{\max} m 2^{l_{\max}} (2^{l_{\max}} + M^2) + n m l_{\max} 2^{l_{\max}}\right), \tag{17}$$

where Θ is the maximum number of iterations needed before convergence, and is a function of $\Theta \approx f(n, m, \max l, w_{\text{solo}}, o_{cv}, D)$; D is the diameter of the agent communication network, defined in Ref. 40 as the maximum of the shortest connection steps between any two agents a_1, a_2. Two agents a_1, a_2 are connected if they interchange information directly in the second phase of the algorithm.

Note that o_{cv} is a time-out parameter of the algorithm for agents that do not find any coalition mate for a particular task. The reader can find more information in Appendix D about the derivation. Intuitively, one can predict a complexity issue related to the exponential term on the maximum number of sensors per task $2^{l_{\max}}$. Indeed, when more sensors per task are added to the simulation, the combination of possibilities in the partial allocation of the sensor subtasks grows worse than exponentially.

In order to reduce the scaling order of the algorithm with the number of sensors, the user may consider reducing the task partial allocation possibilities by eliminating the subtasks with low partiality ($K(j)$).

However, the authors believe this should not be an issue in real-world problems, where a given typical task probably does not need more than three to four different sensors. Furthermore, as mentioned

in Section 1, the future of space systems seems to go in the direction of more distributed architectures with fewer instruments per spacecraft.

Since the value of Θ is hard to calculate in general, Figure 12 shows the number of iterations that the algorithm needs for reaching global consensus in 100 different randomly generated scenarios. We observe that the number of iterations needed to reach convergence is on the order of hundreds for all values of the number of tasks. This number of iterations is proportional to the number of messages interchanged between agents, and thus to communication throughput.

Third, the framework assumes that all agents can always communicate to achieve consensus, and that the costs of communication are negligible. As seen in Figure 12, communication throughput for achieving consensus and conflict-free assignments in market-based decentralized task allocation algorithms can be quite high. This is especially important when simulating space systems, where communication networks are not always strongly connected. Frequent communication requirements can be challenging from both a technological and operational standpoint—although there are simple solutions, such as using communications relay services.

More generally, the current simulation framework does not incorporate several realistic aspects of spacecraft engineering, including orbital dynamics, attitude control, data capacity, link budgets, or realistic division of scores per partially assigned task (e.g., they only depend on the sensors allocated). Other aspects such as docking maneuvers should be carefully modeled, especially issues regarding docking time. We are currently working on integrating these aspects into the framework.

6 | CONCLUSION

In this paper, we have presented a simulation framework for quantifying the value of systems composed of autonomous collaborative agents. Such agents can organize themselves by forming coalitions to perform observational tasks that they could not do on their own. The framework responds to the specific particularities of the Federated Satellite Systems paradigm: opportunism, distributed negotiation, and modularity. By extending the CCBBA, we provide a way to quantify important trade-offs and perform sensitivity analyses in the Federated Satellite System and other similar paradigms. Specifically, this paper has focused on: (1) the effect of time constraints (driven by decorrelation of data sets) on the number of collaborative coalitions formed; (2) the value of distributed systems with coalitions versus monolithic systems under uncertainties related to the failure of sensors and vehicle platforms; and (3) the degradation of performance of the overall system as a function of the number of decision makers (i.e., when there are multiple organizations operating the assets, coalitions that are interesting to all involved parties are harder to find).

Future work will look into developing formal guarantees of performance for the *Modified* CCBBA, incorporating realistic spacecraft engineering aspects into the model (especially astrodynamics), extending the framework for handling dynamic environments in which, for example, new tasks appear during the simulation, exploring analytical game-theoretical models for cross-validation purposes, and analyzing interesting emergent phenomena that could arise from the different topologies of the system of vehicles.

ACKNOWLEDGMENT

The authors would like to thank the CELLEX Foundation for partially funding this research.

ORCID

Ximo Gallud http://orcid.org/0000-0002-1567-9162

REFERENCES

1. Selva D, Dingwall B, Altunc S. A concept for an Agile Mission Development Facility for CubeSat and suborbital missions. In: Proceedings of the 2016 IEEE Aerospace Conference, 2016; Big Sky, MT: IEEE, pages 1–17.
2. Shaw GB. *The Generalized Information Network Analysis Methodology for Distributed Satellite Systems* [Sc.D. thesis]. Cambridge, MA: Massachusetts Institute of Technology; 1999.
3. Jilla C, Miller D. A multiobjective, multidisciplinary design optimization methodology for the conceptual design of distributed satellite systems. In: Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2002; Reston, VA: American Institute of Aeronautics and Astronautics.
4. Hitomi N, Selva D, Blackwell WJ. Exploring the architectural tradespace of severe weather monitoring nanosatellite constellations. AGU Fall Meeting 2014; 2014; San Francisco, CA.
5. Selva D, Crawley EF. Integrated assessment of packaging architectures in earth observing programs. In: Proceedings of the IEEE Aerospace Conference, 2010; Big Sky, MT.
6. Kidder SQ, Kankiewicz JA, Haar THV. The A-Train: how formation flying is transforming remote sensing. In: Proceedings of the The Joint EUMETSAT Conference, 2007; Amsterdam, The Netherlands.
7. Brown O, Eremenko P, Hamilton BA. *Fractionated Space Architectures: a Vision for Responsive Space*. Arlington, VA: Defense Advanced Research Projects Agency; 2006.
8. Golkar A, Lluch ICI. The Federated Satellite Systems paradigm: concept and business case evaluation. *Acta Astronautica*. 2015;111:230–248.
9. Mosleh M, Dalili K, Heydari B. Distributed or monolithic? A computational architecture decision framework. *IEEE Syst J*. 2016. https://doi.org/10.1109/JSYST.2016.2594290.
10. Di L, Moe K, Van Zyl TL. Earth observation sensor web: an overview. *IEEE J Sel Topics Appl Earth Observ Remote Sens*. 2010;3:415–417.
11. Delin KA, Jackson SP. Sensor web: a new instrument concept. In: Proceedings of the Symposium on Integrated Optics, 2001; San Jose, CA: International Society for Optics and Photonics.
12. Chandler PR, Pachter M, Rasmussen S. UAV cooperative control. In: Proceedings of the 2001 American Control Conference (Cat. No.01CH37148), pages 50–55, 2001; Anchorage, Alaska.
13. Sternberg D, Chodas M, Jewison C, Jones M, De Weck O. Multidisciplinary system design optimization of on orbit satellite assembly architectures. In: Proceedings of the IEEE Aerospace Conference, 2015; Big Sky, MT.
14. Heydari B, Dalili K. Emergence of modularity in system of systems: complex networks in heterogeneous environments. *IEEE Syst J*. 2015;9(1):223–231.

15. Ramchurn SD, Polukarov M, Farinelli A, Jennings N, Trong C. Coalition formation with spatial and temporal constraints. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AA-MAS), 2010; Toronto, Canada, pages 1181–1188.
16. Mas-Colell A, Whinston MD, Green JR. *Microeconomic Theory*. New York: Oxford University Press; 1995.
17. Shehory O, Kraus S. Methods for task allocation via agent coalition formation. *Artif Intell*. 1998;101(1–2):165–200.
18. Kraus S. Negotiation and cooperation in multi-agent environments. *Artif Intell*. 1997;94(1–2):79–97.
19. Yun Yeh D. A dynamic programming approach to the complete set partitioning problem. *BIT*. 1986;26(4):467–474.
20. Rahwan T, Michalak T, Jennings NR. Minimum search to establish worst-case guarantees in coalition structure generation. In: Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, 2011; Barcelona, Spain, pages 338–342.
21. Sandholm T, Larson K, Andersson M, Shehory O, Tohmé F. Coalition structure generation with worst case guarantees. *Artif Intell*. 1999;111:209–238.
22. Rahwan T. *Algorithms for Coalition Formation in Multi-Agent Systems* [Ph.D. thesis]. Southampton, England: University of Southampton; 2007.
23. Keinänen H. Simulated annealing for multi-agent coalition formation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Vol. 5559, 2009; Berlin, Heidelberg: Springer, pages 30–39.
24. Vig L, Adams JA. Multi-robot coalition formation. *IEEE Trans Robot*. 2006;22(4):637–649.
25. Manathara JG, Sujit PB, Beard RW. Multiple UAV coalitions for a search and prosecute mission. *J Intell Robot Syst*. 2011;62(1):125–158.
26. Korsah GA, Dias MB, Stentz A. A comprehensive taxonomy for multi-robot task allocation. *Int J Robot Res*. 2013;32:1495–1513.
27. Koes M, Nourbakhsh I, Sycara K. Constraint optimization coordination architecture for search and rescue robotics. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2006; Orlando, FL, pages 3977–3982.
28. Korsah GA. *Exploring Bounded Optimal Coordination for Heterogeneous Teams with Cross-Schedule Dependencies* [Ph.D. thesis]. Pittsburgh, PA: Carnegie Mellon University; 2011.
29. Dias B, Stentz A. A free market architecture for distributed control of a multirobot system. In: Proceedings of the 6th International Conference on Intelligent Autonomous Systems IAS6, 2000; Venice, Italy, pages 115–122.
30. Guerrero J, Oliver G. Multi-robot task allocation strategies using auction-like mechanisms. *Artif Res Dev Front Artif Intell Appl*. 2003;100:111–122.
31. Lin L, Zheng Z. Combinatorial bids based multi-robot task allocation method. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2005; Barcelona, Spain, pages 1145–1150.
32. Augenstein S, Estanislao A, Guere E, Blaes S. Optimal scheduling of a constellation of earth-imaging satellites, for maximal data throughput and efficient human management. In: Proceedings of the International Conference on Automated Planning & Scheduling (ICAPS), 2016; London, UK, pages 345–352.
33. Bianchessi N, Cordeau JF, Desrosiers J, Laporte G, Raymond V. A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *Eur J Oper Res*. 2007;177(2):750–762.
34. Bonnet G, Tessier C. An incremental adaptive organization for a satellite constellation. Organized Adaption in Multi-Agent Systems. First International Workshop, 2009; Estoril, Portugal: Springer-Verlag, pages 108–125.
35. Wolfe WJ, Sorensen SE. Three scheduling algorithms applied to the earth observing systems domain. *Manage Sci*. 2000;46(1):148–166.
36. Wang H, Li X, Liu Y, Zhou B. Summary of intelligent algorithms in planning & scheduling of Earth observation satellite. In: Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, 2010; Xiamen, China: IEEE, pages 480–483.
37. Das S, Wu C, Truszkowski W. Enhanced satellite constellation operations via distributed planning and scheduling. In: Proceedings of the International Symposium on Artificial Intelligence and Robotics & Automation in Space, 2001; Quebec, Canada.
38. Van Der Horst J, Noble J. Task allocation in dynamic networks of satellites. IJCAI Workshop on Artificial Intelligence in Space, 2011; Barcelona, Catalonia.
39. Whitten AK, Choi HL, Johnson LB, How JP. Decentralized task allocation with coupled constraints in complex missions. In: Proceedings of the American Control Conference, 2010; Baltimore, MD.
40. Choi HL, Brunet L, How JP. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans Robot*. 2009;25(4):912–926.
41. Choi HL, Kim KS, Johnson LB, How JP. Potential game-theoretic analysis of a market-based decentralized task allocation algorithm. In: Proceedings of the 12th International Symposium, 2016.
42. Johnson L, Ponda S, Choi HL, How J. Asynchronous decentralized task allocation for dynamic environments. In: Proceedings of the AIAA Infotech@Aerospace Conference, 2011; St. Louis, MO.
43. Hunt S, Meng Q, Hinde C, Huang T. A consensus-based grouping algorithm for multi-agent cooperative task allocation with complex requirements. *Cogn Comput*. 2014;6(3):338–350.
44. Argyle ME, Beard RW, Casbeer DW. Multi-team Consensus Bundle Algorithm. In: Valavanis K, Vachtsevanos G, eds. *Handbook of Unmanned Aerial Vehicles*. Dordrecht: Springer; 2015.
45. Butler Z, Hays J. Task allocation for reconfigurable teams. *Robot Autonomous Syst*. 2013;68(C):59–71.
46. Whitten A. *Decentralized Planning for Autonomous Agents Cooperating in Complex Missions* [S.M. thesis]. Cambridge, MA: Massachusetts Institute of Technology; 2010.
47. Michel F, Beurier G, Ferber J. The TurtleKit Simulation Platform: Application to Complex Systems. SITIS: Signal-Image Technology and Internet-Based Systems, 2005, Yaounde, Cameroon. Proceedings of the 1st International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2005, November 27 - December 1, 2005, Yaounde, Cameroon., 2005.
48. Box GEP, Hunter JS, Hunter WG. *Statistics for Experimenters: Design, Innovation, and Discovery*. 2nd ed. Hoboken, NJ: Wiley; 2005.
49. Ponda SS, Johnson LB, Geramifard A, How JP. Cooperative Mission Planning for Multi-UAV Teams. In: Valavanis K, Vachtsevanos G, eds. *Handbook of Unmanned Aerial Vehicles*. Dordrecht: Springer; 2015.
50. Tucker MJ. The decorrelation time of microwave radar echoes from the sea surface. *Int J Remote Sens*. 1985;6(7):1075–1089.
51. O'Neill MG. Assessing the impacts of fractionation on pointing-intensive spacecraft. In: Proceedings of the AIAA Space 2009 Conference and Exposition, 2010; Cambridge, MA, page 185.
52. Ponda S, Redding J, Choi H-L, How JP, Vavrina M, Vian J. Decentralized planning for complex missions with dynamic communication constraints. In: Proceedings of the 2010 American Control Conference, 2010; Baltimore, MD: IEEE, pages 3998–4003.

AUTHORS' BIOGRAPHIES

Ximo Gallud holds a Research Assistant position at the Space Propulsion Laboratory (SPL), MIT. Before joining the SPL, Ximo was a Research Assistant at the Systems Engineering, Architecture and Knowledge Lab, Cornell University. In October 2016, he obtained a B.Sc. degree in Aerospace Engineering Technology and a B.Sc. degree in Engineering Physics from the Universitat Politecnica de Catalunya (UPC), Barcelona, Spain. His research interests focus on miniaturization of space components and the coordination of satellites with limited capability constraints. Ximo Gallud has significant experience in nanosatellite design and physics of electromagnetic traps obtained during research internships in the Remote Systems Laboratory (UPC) and the Institut de Ciencies Fotoniques de Castelldefels (ICFO).

Daniel Selva received a Ph.D. in Space Systems from MIT in 2012, and he is an Assistant Professor at the Sibley School of Mechanical and Aerospace Engineering and in the Systems program at Cornell University, where he directs the Systems Engineering, Architecture, and Knowledge (SEAK) Lab. His research interests focus on the application of knowledge engineering, global optimization, and machine learning techniques to systems engineering, design, and architecture, with a strong focus on space systems. Prior to MIT, Daniel worked for 4 years in Kourou (French Guiana) as an avionics specialist within the Ariane 5 Launch team. Daniel has a dual background in electrical engineering and aeronautical engineering, with degrees from Universitat Politecnica de Catalunya in Barcelona, Spain, and Supaero in Toulouse, France. He is also a Faculty Fellow at the Mario Einaudi Center for International Studies, and a member of the AIAA Intelligent Systems Technical Committee.

How to cite this article: Gallud X, Selva D. Agent-based simulation framework and consensus algorithm for observing systems with adaptive modularity. *System Engineering*. 2018;21:432–454. https://doi.org/10.1002/sys.21433

APPENDIX A: HIGH-LEVEL DESCRIPTION OF THE SIMULATION FRAMEWORK AND *MODIFIED* CCBBA ALGORITHM

The architecture of the multiagent simulation framework presented in this paper is summarized in the state machine diagram and functional flow diagram shown in Figures 9 and 10. This definition of agent behavior based on state machine diagrams is particular for the library we used for building the framework, TurtleKit.[47]

In the active state, each supplier agent *wakes up* and requests an *active* role.

In the *Request Task Map* state, each agent sends a request to the Scenario for new available tasks. If the agent has not received any subtask or it considers that it cannot do any of them because it does not have the appropriate sensor to do them, then the agent stops performing tasks. Note that the decision of an agent to stop may be the best available alternative if none of the remaining tasks are doable or interesting, especially since no new tasks may appear over time in this framework. In the *Planning State*, each agent executes the *Modified* CCBBA algorithm (shown in Figure 10). The *Waiting phase* has been added to achieve synchrony among agents. Indeed, since communication between agents is an essential step in the consensus phase of the algorithm, agents must wait for each other to finalize their planning phases.

APPENDIX B: TOY EXAMPLE

This appendix walks the reader through a toy example (Figure 11) to illustrate how the algorithm works.

B.1 Description of the example

In this toy example we have two agents, $\boldsymbol{A} = \{1, 2\}$, carrying an infrared and a microwave sensor, respectively, $\epsilon_1 = \{IR\}, \epsilon_2 = \{MW\}$. There are two tasks, $\boldsymbol{V} = \{A, B\}$. Agents can travel at a maximum velocity of $vel_{\max} = 1$.

The parameters of the tasks are the following:

$$\begin{cases} S = S_{\max}, & \text{if agent contains sensor of } j \\ 0 & \text{otherwise,} \end{cases} \tag{B1}$$

$c_v = 2.0$.

The superadditive score function is as follows:

$$\alpha\left(\frac{K(j)}{l}\right) = \begin{cases} 1, & \text{if } \left(\frac{K(j)}{l}\right) = 1 \\ \frac{1}{3} & \text{otherwise.} \end{cases}$$

Task positions: $\boldsymbol{x_A} = (0, 3), \boldsymbol{x_B} = (4, 3)$.

Set of subtasks per task: $\boldsymbol{J_A} = [\{IR\}_{\{MW\}}, \{MW\}_{\{IR\}}, \{IR\}_{\{\}}, \{MW\}_{\{\}}]$. $\boldsymbol{J_B} = [\{MW\}_{\{\}}]$.

The aforementioned subtasks will be labeled as $\boldsymbol{J_A} = \{j_{A1}, j_{A2}, j_{A3}, j_{A4}\}$, and $\boldsymbol{J_B} = \{j_{B1}\}$.

It can be inferred that $l_A = 2$, $l_B = 1$, $K(j_{A1}) = K(j_{A2}) = 2$, $K(j_{A3}) = K(j_{A4}) = K(j_{B1}) = K(j_{A4}) = 1$. $\mathbf{D_A}$ and $\mathbf{T_A}$ matrices are the same as the ones shown in Section 2.2.

The planning horizon for this example is $M = 2$, thus, each agent will select a maximum of two tasks to perform. As the agents only have one sensor, the maximum of tasks will also correspond to the maximum of subtasks. The notation we use for defining the bids for these two subtasks that agent 1 can place are $c_1^{j\,(1)}, c_1^{j\,(2)}$, where the exponents 1 and 2 indicate the position in the bundle. Idem for agent 2, $c_2^{j\,(1)}, c_2^{j\,(2)}$.

B.2 Overview of the algorithm

In the first phase of the algorithm, each agent will calculate a bid on the $M = 2$ subtasks that maximize their utility. If they can outbid the

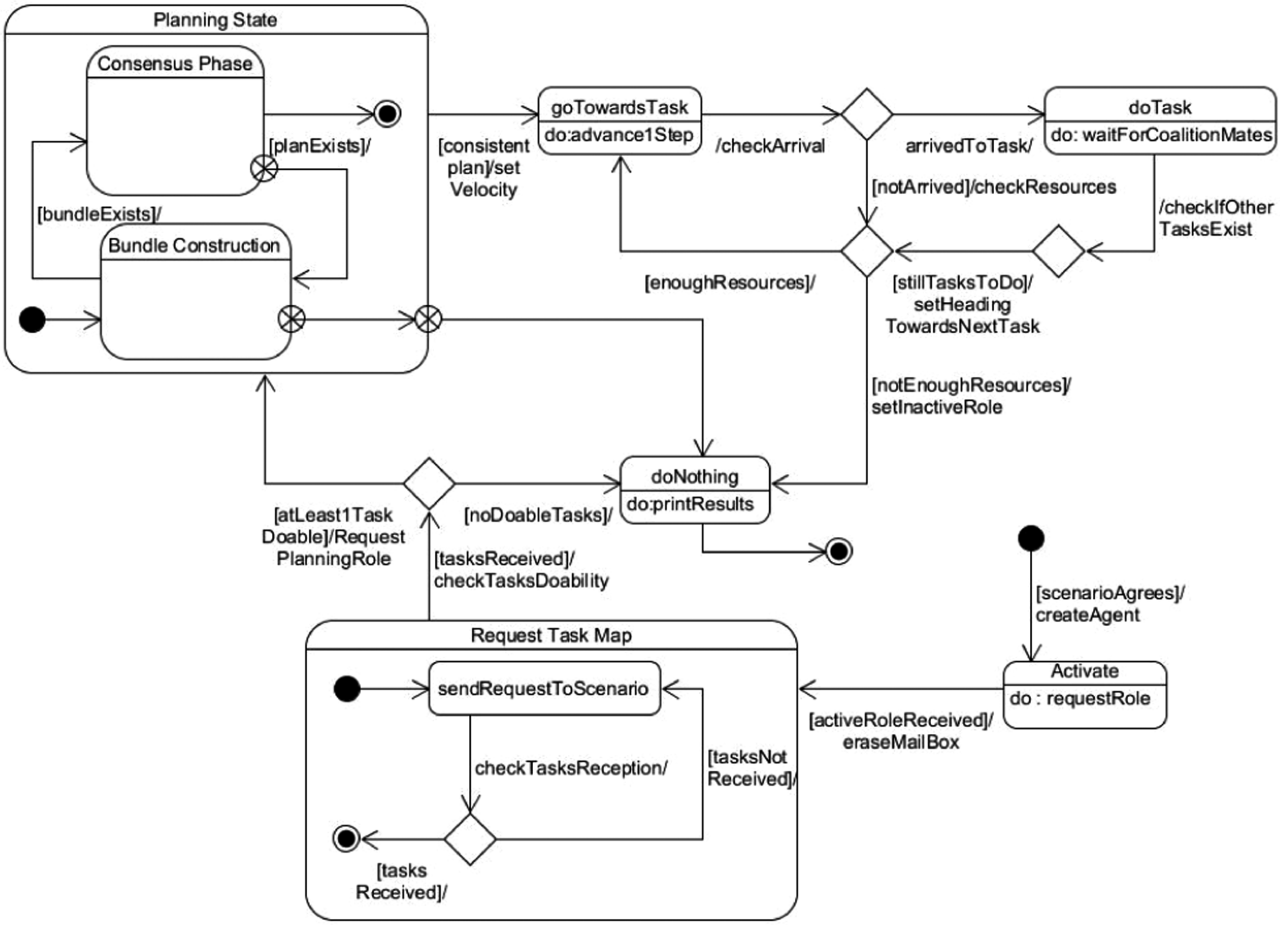

FIGURE 9 State machine diagram of the behavior of a supplier agent

winner bids for these subtasks, they will update the vectors of information from other agentâs bids and winners, $\mathbf{y_a}, \mathbf{z_a}, \mathbf{tz_a}, \mathbf{s_a}$ with their winning bids and posting themselves as winners. They will store the information of all subtasks in the following order $\{j_{A1}, j_{A2}, j_{A3}, j_{A4}, j_{B1}\}$. For example, if agent 1 thought the winner of task $j_{A2} = \{MW\}_{\{IR\}}$ is 2, then the vector component $\mathbf{z_1} = \{\ldots, 2, \ldots, \ldots, \ldots\}$

During the second phase, the agents interchange their information variables with the updated information of the bids they have placed in the phase 1. During this phase, agents may release the some tasks from their bundle and their path, as they enter in conflict with the other agent's schedule. Task release is governed by the set of rules described in Ref. 40 with our modifications. After that, they will have to go again to the first phase to reconstruct the bids with the updated information ($\mathbf{y_a}, \mathbf{z_a}, \mathbf{tz_a}, \mathbf{s_a}$) obtained from the message interchange of other agents in the second phase. This process is repeated iteratively until consensus is reached and each agent performs a conflict-free schedule.

B.3 Numerical implementation

The following paragraphs walk the reader through the details of the algorithm, step by step.

Iteration 1: During the first phase of the algorithm, at the first iteration, $\zeta = 0$, the vectors of information per agent are the following: Winner's vector: $\mathbf{z_1} = \mathbf{z_2} = \{\emptyset, \emptyset, \emptyset, \emptyset, \emptyset\}$. Maximum bids vector: $\mathbf{y_1} = \mathbf{y_2} = \{0, 0, 0, 0, 0\}$.

The arrival times vector $\mathbf{tz_a}$ stores the time at which each subtask is starting, according to a: $\mathbf{tz_1} = \mathbf{tz_2} = \{0, 0, 0, 0, 0\}$. Agent 1 calculates the two bids for the subtasks that will maximize its utility, according to Algorithm 1.

Initially, agent 1 will choose the highest bid between the bids placed for $\{j_{A1}, j_{A4}\}$, as these are the only tasks with the sensors of 1.

Even though subtask $\{j_{A1}\}$ needs a coalition mate (i.e., subtask $\{j_{A2}\}$ must also be allocated, as $\mathbf{D}(j_{A1}, j_{A2}) = 1$), the algorithm allows agents to place bids for these tasks without assuring a coalition mate a specific number of times. This is called the optimistic strategy in the CCBBA[39] (see Section 2.2).

Agent 1 first calculates where to do the subtasks, this is $\mathbf{x}$, and the time of arrival, $t_1^{A1} = t_1^{A4} = t_1^A$. As 1 still has not exchanged information with the other mate, 1 will decide where to perform the task and at what time without considering 2 schedule. As stated by the score function agent 1 decides to perform the observation subtask at the location of the task, thus $\mathbf{x_1^{A1}} = \mathbf{x_1^{A4}} = \mathbf{x_A}$.

Agent 1 decides to arrive to the task at its earliest convenience, thus, if the maximum velocity of agent 1 is 1, it decides to arrive to A at $t_1^A = 3$. The reader can find more details on how the agents choose the starting times as a function of their mates schedules.[39]

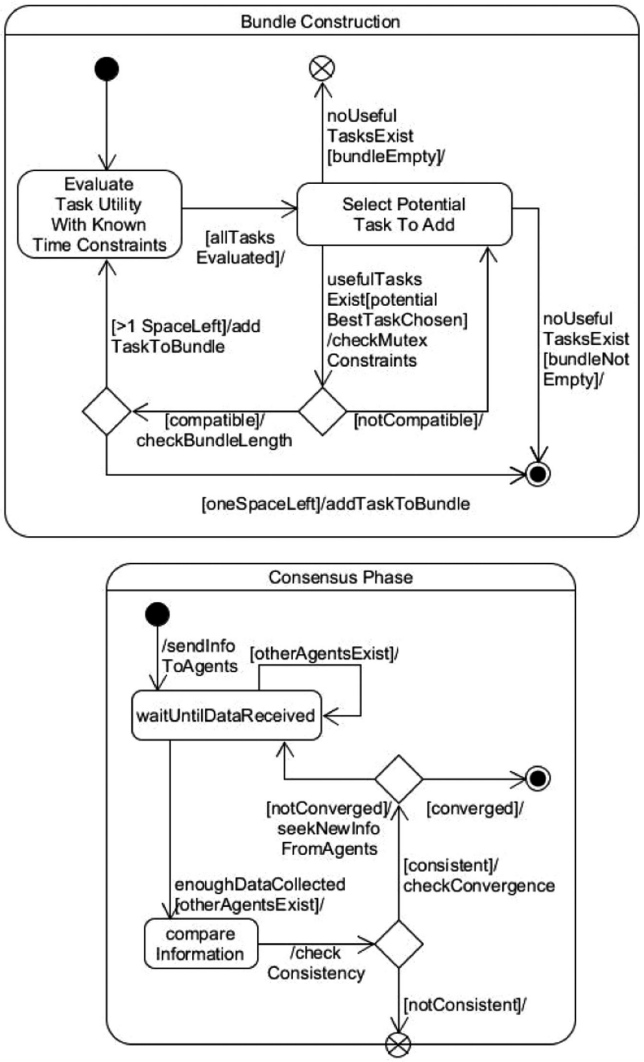

FIGURE 10 Functional flow diagram for an agent in the bundle construction phase (first phase of the *Modified* CCBBA) and the consensus phase (second phase of the *Modified* CCBBA

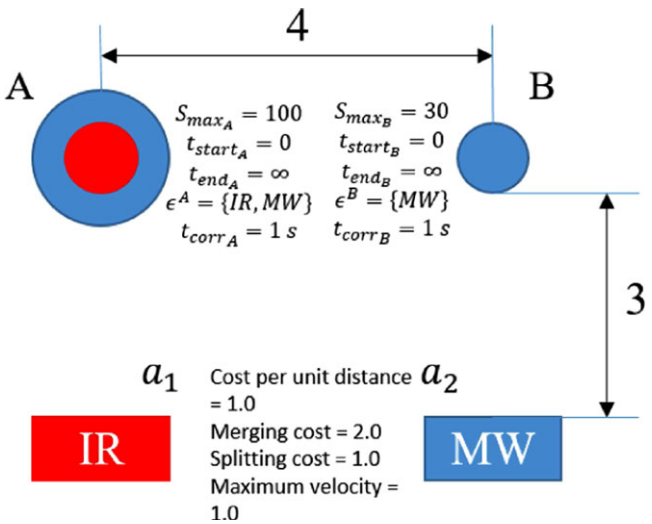

FIGURE 11 Toy example initial scenario

The path costs g are $g_A = 3$. If no coalition mate is still known for the task ($\mathbf{z_1}$ is empty), the coalition cost is 0. Thus, $p_A = 0$. The cost of taking an image, c_A is 2. In total, the bid that agent 1 places for the subtask j_{A1}, $c_1^{A1(1)} = 100\alpha(\frac{2}{2})\frac{1}{2} - 3 - 0 - 2 = 45.0$.

Similarly, agent 1 will bid for subtask j_{A4}.

$$c_1^{A1(1)} = 100\alpha\left(\frac{1}{2}\right)\frac{1}{1} - 3 - 0 - 2 = 27.33.$$

Agent 1 chooses j_{A1} as a part of its bundle. Thus, the information vectors that it will share with agent 2 are: $\mathbf{tz_1} = \{3.0, 0, 0, 0, 0\}$, $\mathbf{y_1} = \{45.0, 0, 0, 0, 0\}$, $\mathbf{z_1} = \mathbf{z_2} = \{1, \emptyset, \emptyset, \emptyset, \emptyset\}$, $b_1 = \{j_{A1}\}$.

Initially, agent 2 does not know what agent 1 has bidden yet, so it follows the same process when bidding for subtasks j_{A2}, j_{A4}, j_{B1}. Agent 2's bids are as follows: $c_2^{A2(1)} = 43.0$, $c_2^{A4(1)} = 27.33$, $c_2^{B1(1)} = 27.0$.

Thus, agent 2 selects the first subtask of its bundle, and places a time of arrival.

$$\mathbf{tz_2} = \{0, 5.0, 0, 0, 0\}, \quad \mathbf{y_2} = \{0, 43.0, 0, 0, 0\}, \quad \mathbf{z_2} = \{\emptyset, 2, \emptyset, \emptyset, \emptyset\},$$

$$\mathbf{b_2} = \{j_{A2}\}.$$

Agent 2 repeats the process for the rest of the tasks. As done in Refs. 39 and 40, the next bids values are the differences between the total accumulated utility after adding the new task in a feasible position and the total accumulated utility before adding that particular task.

Agent 2 would only be able to place a bid on a single subtask for a particular task, thus $c_2^{A4(2)} = 0$.

For calculating $c_2^{B1(2)}$, agent 2 calculates the differences in utility for all the possible paths containing the task that has already incorporated to its bundle, j_{A2}, and j_{B1}. Agent 2 will then choose the path that maximizes this difference and satisfies the constraints imposed by the times of arrival. For example, the path $[j_{B1}, j_{A2}]$ will be more valuable than the path $[j_{A2}, j_{B1}]$, but agent 2 has already agreed to do task j_{A2} a time $t_2^{A2} = 5.0$. Thus, even if traveling at maximum velocity, the otherwise optimal path is discarded.

Finally, after computing the bids: $\mathbf{b_2} = \{j_{A2}, j_{B1}\}$, $\mathbf{tz_2} = \{0, 5.0, 0, 0, 9.0\}$, $\mathbf{y_2} = \{0, 43.0, 0, 0, 24.0\}$, $\mathbf{z_2} = \{\emptyset, 2, \emptyset, \emptyset, 2\}$.

During the second phase, the agents interchange their information variables. Agent 1 realizes that the time of arrival to task j_{A1} does not satisfy the time constraints.

Thus, agent 1 is forced to release its task j_{A1}, as $t_1^{A1} - t_2^{A3} = 2.0 > t_{\text{corr}} = 1$. During this process, agent 1 will decrease by 1 its number of allowed opportunities for bidding on a task with dependencies (w_{solo}). After releasing the task, the information vectors of 1 are: $\mathbf{tz_1} = \{0, 5.0, 0, 0, 9.0\}$, $\mathbf{y_1} = \{0, 43.0, 0, 0, 24.0\}$, $\mathbf{z_1} = \{\emptyset, 2, \emptyset, \emptyset, 2\}$.

During the second phase of the algorithm, agent 2 has not found any conflict, therefore it keeps its vectors of information.

Iteration 2: During iteration 2, $\zeta = 1$, agent 1 bids on task j_{A1} again.

However, as noticed by its winners vector, $\mathbf{z_1}$, agent 1 now knows that it has a coalition mate that has won task j_{A2}, 2. Agent 1 will have to deal with the coalition costs associated to merging into a coalition, so the bid now for j_{A1} is $c_1^{A1(1)} = 43.0$, which is the same as in the previous iteration minus the merging cost. Agent 1 assumes that the score awarded for the allocation of a task can diminish with the arrival time.

Thus, agent 1 will set a starting time of j_{A1} that fulfills the decorrelation time constraints and provides itself with the maximum possible score (that is, arriving there at the earliest time possible $t_1^{A1} = 4.0$). After the first phase of the second iteration, the information vectors are the following: $\mathbf{tz_1} = \{4.0, 5.0, 0, 0, 9.0\}$, $\mathbf{y_1} = \{43.0, 43.0, 0, 0, 24.0\}$, $\mathbf{z_1} = \{1, 2, \emptyset, \emptyset, 2\}$.

Agent 2 does not do anything in the first phase of the second iteration, as it has already reached the maximum number of tasks allowed by the planning horizon. During the second phase of the second iteration, agent 1 will not see any incompatibility with the vectors from 2, as the bidding information from 2 has not changed, and agent 1 has already adapted its arrival time vector to the one from 2. Agent 1 will update its coalition mates matrix $\Omega_1 = [2 \quad 0]$. However, agent 2 will notice that agent 1 is able to make a coalition for task j_{A2}. Agent 2 had not previously accounted for the merging costs when it was bidding for j_{A2}. Agent 2 will thus release the task j_{A2}. Furthermore, as the bid placed by 2 for task j_{B1}, $c_2^{B1(2)}$, depends on $c_2^{A2(1)}$, agent 2 will have to release both tasks. As the releases of 2 are done because of coalition cost miscalculation, agent 2 does not communicate these updates to agent 1.

Its information vectors will be: $\mathbf{tz_2} = \{4.0, 0, 0, 0, 0\}$, $\mathbf{y_2} = \{43.0, 0, 0, 0, 0\}$, $\mathbf{z_2} = \{1, \emptyset, \emptyset, \emptyset, \emptyset\}$.

Iteration 3 ($\zeta = 2$): During the first phase of the third iteration, agent 1 considers that its bids agree with the constraints, so it keeps all its bids. Thus, no changes are made to the vectors of 1. Agent 2 recalculates its bids according to the coalition mate structure. The new bids account for the merging cost of subtask j_{A2} and the cost of breaking the coalition for attending subtask j_{B1}. Thus, in the third iteration, the new information vectors will be: $\mathbf{tz_2} = \{4.0, 5.0, 0, 0, 9.0\}$, $\mathbf{y_2} = \{43.0, 41.0, 0, 0, 23.0\}$, $\mathbf{z_2} = \{1, 2, \emptyset, \emptyset, 2\}$, The updated coalition mates matrix will be $\Omega_2 = [1 \quad \emptyset]$.

In the second phase of the third iteration, both agents will realize that their bidding information is consistent and all the constraints are met. The algorithm has thus converged and a conflict-free plan has been achieved.

APPENDIX C: FORMULATION AS A MIXED INTEGER QUADRATIC PROBLEM

The formulation (objectives and constraints) used for solving the centralized XD [ST-MR-TA] problem with engagement costs is provided below. The formulation extends the one described in Korsah[28] to agents carrying more than one sensor ($|\epsilon_a| > 1$). The reader can refer to Tables 2 and 3 for a description of the variables and constants used in this formulation.

TABLE 2 Variables of the MIQP formulation of the XD [ST-MR-TA] problem with engagement costs

Variable	Description	Type
x_a^π	Whether agent a performs path π.	Binary
d_a^j	Waiting time of agent a before starting subtask j.	Real
t_j	Execution time for j.	Real
$\beta_{aa'}^i$	Whether a' is a coalition mate of a at the subtask ordered in the position i of the path of a.	Binary

TABLE 3 Constants of the MIQP formulation of the XD[ST-MR-TA] problem with engagement costs

Constant	Description	Type
S_j	Score that subtask j is worth.	Real
c_v	Intrinsic cost of task v.	Real
g_a^π	Total travel cost of path π.	Real
λ_j	Linear delay discount.	Real
$y_a^{j\pi}$	Whether subtask j is included in the path $\pi \in \pi_a$.	Binary
$\gamma_{\pi a}^{si}$	Whether subtask in the position i of the path π of agent a needs a coalition mate with sensor $s \in \epsilon$.	Binary
$\delta_a^{jj'\pi}$	Whether subtask j and j' are included on the path $\pi \in \pi_a$ and j' is addressed at some time before j.	Binary
$t_a^{\min_{j\pi}}$	Time that agent a would arrive at the location of subtask j in the path $\pi \in \mathbf{p}_a$ if a did not wait for any coalition mate.	Real
c_m	Merging cost.	Real
c_s	Splitting cost.	Real

Maximize:

$$\sum_{j\in J_v\times V}\sum_{a\in A}\sum_{\pi\in\pi_a}(S_j - c_v)y_a^{j\pi}x_a^\pi - \sum_{a\in A}\sum_{\pi\in\pi_a} g_a^\pi x_a^\pi$$
$$- \sum_{j\in J_v\times V} S_j\lambda_j t_j - \sum_{i=1}^{i=M}\sum_{a\in A}\sum_{a'\in A, a'\neq a} c_m\beta_{aa'}^i$$
$$+ \sum_{i=2}^{i=M}\sum_{a\in A}\sum_{a'\in A, a\neq a'} c_m\beta_{aa'}^i\beta_{aa'}^{i-1} - \sum_{i=1}^{i=M-1}\sum_{a\in A}\sum_{a'\in A, a\neq a'} c_s\beta_{aa'}^i$$
$$+ \sum_{i=2}^{i=M}\sum_{a\in A}\sum_{a'\in A, a\neq a'} c_s\beta_{aa'}^i\beta_{aa'}^{i-1}.$$

Subject to:

$$\sum_{\pi\in\pi_a} x_a^\pi \leq 1 \qquad \forall a \in \mathbf{A}, \tag{C1}$$

$$\sum_{j\in J_v\times V}\sum_{\pi\in\pi_a} y_a^{j\pi}x_a^\pi \leq 1 \qquad \forall j \in \mathbf{J_v}\times\mathbf{V}, \tag{C2}$$

$$t_j - \sum_{a\in A}\sum_{\pi\in\pi_a} t_a^{\min_{j\pi}}\pi_a^{j\pi}x_a^\pi - \delta_a^{jj'\pi}d_a^j x_a^\pi = 0 \qquad \forall j', j \in J_v\times\mathbf{V},$$
$$j' \neq j, \tag{C3}$$

$$\beta_{aa'}^i = 0 \qquad \forall a, a' \in \mathbf{A} \text{ such that } \epsilon_a = \epsilon_{a'}, \forall i \tag{C4}$$

$$\sum_{a\in A, a\neq a', \epsilon_a = s}\beta_{aa'}^i = \sum_{\pi\in\pi_a}\gamma_{\pi a}^{si}x_a^\pi \qquad \forall a' \in \mathbf{A},$$
$$\forall s \in \epsilon \text{ such that}$$
$$\epsilon_{a'} \neq s, \forall i, \tag{C5}$$

$$\sum_{i=1}^{i=M} \beta_{aa'}^{i} = \sum_{i=1}^{i=M} \beta_{aa'}^{i} \qquad \forall a, a' \in \mathbf{A}, a \neq a', \tag{C6}$$

$$\sum_{a \in \mathbf{A}} \sum_{\pi \in \pi_a} y_a^{j\pi} x_a^{\pi} = \sum_{a' \in \mathbf{A}} \sum_{\pi \in \pi_{a'}} y_{a'}^{j'\pi} x_{a'}^{\pi} \qquad \forall j, j' \text{ such that } \mathbf{D}(j, j') = 1, \tag{C7}$$

$$t_j - t_{j'} \leq t_{\text{corr}} \qquad \forall j, j' \text{ such that } \mathbf{D}(j, j') = 1, \tag{C8}$$

$$t_j - t_{j'} \geq -t_{\text{corr}} \qquad \forall j, j' \text{ such that } \mathbf{D}(j, j') = 1. \tag{C9}$$

Constraint C1 assigns one route per agent as a maximum. Constraint C2 prevents a subtask from being assigned to multiple agents. Constraint C3 ensures a correct arrival time to the subtasks. Constraint C4 prevents agents with the same sensor from participating in coalitions, in a similar way to CoalitionTest. Constraint C5 ensures coalition mates of a particular sensor for dependent subtasks in a specific bundle position l. Constraint C6 ensures reciprocity of coalition formation, that is, if agent k has a coalition mate a, then agent a must have also k as its coalition mate. Equation C7 models subtask dependencies. Equations C8 and C9 model time decorrelation constraints.

APPENDIX D: ESTIMATION OF COMPLEXITY

This appendix derives the complexity formula shown in Equation 17.

In the worst-case scenario, we assume that all the tasks need the maximum number of sensors, that is, $l_{\max} = \max l,\ \forall v \in \boldsymbol{V}$. In this case, the total number of subtasks per task, N_{sub_v}, which are depicted in Equation 2 turns into $l_{\max} 2^{l_{\max}-1}$. Thus, the total number of subtasks each agent has to compute a bid for is $m l_{\max} 2^{l_{\max}-1}$. The *Modified* CCBBA runs in two phases and keeps iterating until a consensus has been reached, thus the overall complexity is given by:

$$\Theta \cdot O(\text{First phase} + \text{Second phase}). \tag{D1}$$

In the worst case, the first phase of the algorithm builds the bundle from scratch. Thus, for each subtask added to the bundle, agents will have had to place a bid before:

$$\begin{aligned} O(\text{First Phase}) \approx O(M \cdot (m l_{\max} 2^{l_{\max}} \cdot & \\ (\text{CalculateBidsForSubtask} & \\ + \text{CoalitionTest} + \text{MutexTest}) & \\ + O(\text{Further calculations})), & \end{aligned} \tag{D2}$$

where:

$$O(\text{CalculateBidsForSubtask}) \approx l_{\max} + M^2.$$

The latter equation contains the order of the functions used for getting the minimum and maximum time of arrival to a specific subtask,[39]

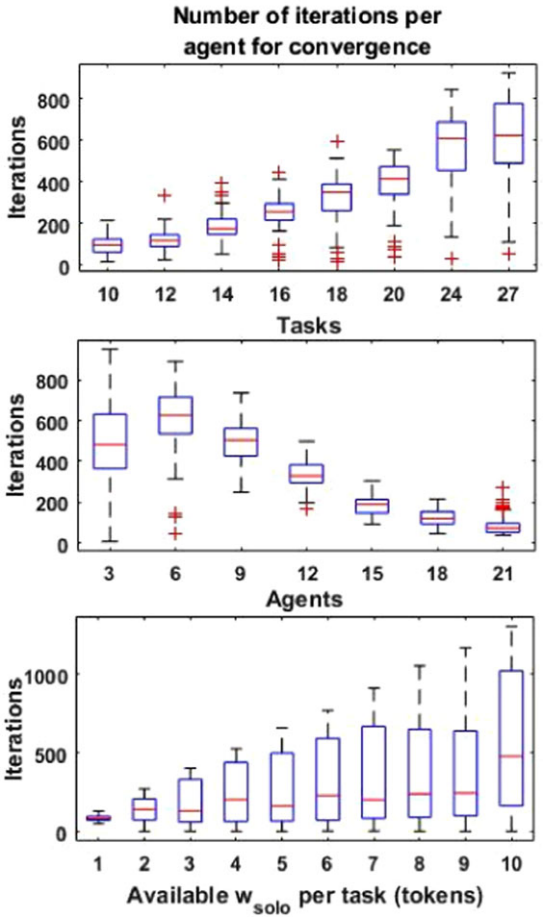

FIGURE 12 Boxplot of the number of iterations Θ of the *Modified* CCBBA algorithm as a function of the number of tasks (top), agents (middle) and w_{solo} (bottom)

which is proportional to the maximum number of dependent subtasks (i.e., one per sensor, $l_{\max}$). The computation of the paths scales with M. As in Ref. 52, not all possible paths are computed (this would scale with $M!$). Instead, the new subtask j to be added to the bundle is added in any position of the existing bundle, without changing the relative positions of the other subtasks. The computation of the utility for each path scales with M. Thus, the combined computation of the path and the utility scales with M^2.

$$O(\text{CoalitionTest}) \approx l_{\max},$$

$$O(\text{MutexTest}) \approx l_{\max} 2^{l_{\max}}.$$

CoalitionTest sums the bids of all the dependent subtasks ($l_{\max}$). MutexTest sums the bids of all the dependent subtasks ($l_{\max}$) and compares it with all the mutexed summations. Thus, it scales with the total number of subtasks per task $l_{\max} 2^{l_{\max}}$. Further calculations include (mostly) choosing a subtask to place a bid for ($O(m l_{\max} 2^{l_{\max}})$). This is of lower order than the latter term of the sum.

The second phase compares information of the bids placed by all the agents in all the subtasks. Thus, $O(\text{Second Phase}) \approx O(n m l_{\max} 2^{l_{\max}})$.

The maximum number of iterations, Θ, has not been extracted yet. The authors of the original CBBA[40] provided a number for

the maximum number of iterations needed before convergence, $D \times \max(Mn, m)$, where D corresponds to the maximum distance in the communication matrix existent between all the agents. Because in our algorithm, agents can release tasks even if they have not been outbid, this guarantee of convergence does not apply. Figure 12 provides an empirical analysis of the number of iterations needed for convergence as a function of n, w_{solo}, m. Note that w_{solo} responds to the total number of times that an agent is able to bid *optimistically* on a dependent subtask without meeting any of the constraints.

APPENDIX E: CODE AND VIDEO

The source code and a video example of this simulation framework can be found in the address below: https://github.com/ximogallud/MCBBA.