MASSpace 2026

Proceedings of the
2nd International Workshop on
Multi-Agent Systems in Space (MASSpace 2026)

Edited by

Itai Zilberstein, Carnegie Mellon University
Gauthier Picard, ONERA, France
Steve Chien, Jet Propulsion Laboratory, California Institute of Technology, USA
26 May 2026, Paphos, Cyprus

Preface

This volume contains the papers presented at MASSpace 2026: The 2nd International Workshop on Multi-Agent Systems for Space was held on 26th May 2026 in Paphos Cyprus as part of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) 2026. This workshop is the 2nd in a series with the 1st MASSpace at AAMAS in Auckland, NZ (2024).

The volume includes 1 invited talk abstract, 1 invited talk paper, and 12 reviewed full papers. The proceedings and select presentations will be archived at the MASSpace Github site below.

https://mas-space.github.io/aamas2026ws/.

12 May 2026 Pasadena, CA

Itai Zilberstein, Carnegie Mellon University

Gauthier Picard, ONERA, France

Steve Chien, Jet Propulsion Laboratory, California Institute of Technology, USA

Organizers

Itai Zilberstein, Carnegie Mellon University

Gauthier Picard, ONERA, France

Steve Chien, Jet Propulsion Laboratory, California Institute of Technology, USA

Program Committee

Caleb A. Adams, NASA Ames Research Center

Alexandre Albore, ONERA

Abigail Breitfeld, Carnegie Mellon University

Thibault Gateau, ISAE-SUPAERO

Alessandro Golkar, Technical University of Munich

Tal Grinshpoun, Ariel University

Jonathan Guerra, Airbus Defence & Space

Christopher Hays, United States Air Force Research Laboratory

Michael J Iatauro, NASA Ames Research Center

Elsy Kaddoum, IRIT

Nicolas Longepe, ESA

Cedric Pralet, ONERA

Serge Rainjonneau, Thales Alenia Space

Ananya Rao, Carnegie Mellon University

David Rijlaarsdam, Ubotica Technologies

Federico Rossi, Jet Propulsion Laboratory, California Institute of Technology

Detailed Program

08:45-10:15		**Session 1**	
	08:45-09:35	Invited Talk: *Developing Distributed Autonomous Space Systems*	Caleb Adams (NASA Ames)
	09:35-09:55	*Large-Scale Continual Scheduling and Execution for Dynamic Distributed Satellite Constellation Observation Allocation*	Itai Zilberstein (Carnegie Mellon University), Steve Chien (Jet Propulsion Laboratory)
	09:55-10:15	*The Role of Central Nodes in Multi-Task Allocation for Distributed Space-Based Observation Systems*	Vincenzo Messina (Technical University of Munich), Marco Camporeale (Technical University of Munich), Rob Vingerhoeds (Fédération FONISEN, Université de Toulouse), Alessandro Golkar (Technical University of Munich)
10:15-11:00		**Morning Coffee Break**	
11:00-12:20		**Session 2**	
	11:00-11:20	*Dynamic Tool Generation for Autonomous Multi-Agent Systems in Space Missions*	Dominik Opitz (DLR) Carsten Hartmann (DLR-GSOC) Tobias Hecking (DLR) Michael Felderer (DLR)
	11:20-11:40	*Autonomous Federation of Earth Observation Constellations: An End-to-End Demonstrator based on the DOMINO Architecture*	Cyrille De Lussy (Airbus Defence and Space) Stéphane Derrien (Cap Gemini) Marie Devant (Cap Gemini) Jean-Loup Farges (DTIS, ONERA, Université de Toulouse) Jonathan Guerra (Airbus Defence and Space) Philippe Pavero (Airbus Defence and Space) Gauthier Picard (DTIS, ONERA, Université de Toulouse) Cédric Pralet (DTIS, ONERA, Université de Toulouse) Jakub Rezler (ITTI) Raivis Skadiņš (Tilde)
	11:40-12:00	*Solving Organizational Multi-Robot Task Allocation Problems with Consensus-based Auctions*	Victor Guillet (DTIS, ONERA, Université de Toulouse) Christophe Grand (DTIS, ONERA, Université de Toulouse) Charles Lesire (DTIS, ONERA, Université de Toulouse) Gauthier Picard (DTIS, ONERA, Université de Toulouse)
	12:00-12:20	*Multi-Satellite Observation Tasks Dispatching and Scheduling*	Romain Barrault (DTIS, ONERA, Université de Toulouse) Cedric Pralet (DTIS, ONERA, Université de Toulouse) Gauthier Picard (DTIS, ONERA, Université de Toulouse) Eric Sawyer (CNES, Université de Toulouse)
12:30-14:00		**Lunch Break**	
14:00-15:30		**Session 3**	
	14:00-14:50	Invited Talk: *A Game-Theoretic Framework for Distributed Fallback Autonomy in Satellite Mega-Constellations*	Daniel Reynolds (MIT)
	14:50-15:10	*Satellite scheduling optimization for BepiColombo STC channel target-phase*	Alberto Avallone (University of Modena and Reggio Emilia) Francesco Sala (University of Modena and Reggio Emilia) Manuel Iori (University of Modena and Reggio Emilia)
	15:10-15:30	*Toward Automated Operational Assist in Satellite Fleet Management via Organizational MARL*	Julien Soulé (University of Luxembourg)
15:30-16:15		**Afternoon Coffee Break**	
16:15-17:35		**Session 4**	
	16:15-16:35	*Decentralized Dynamic Task Allocation under Limited Communication Range*	Alexandre Kha (Thales CortAIx-Labs) Mathieu Marchand (Thales CortAIx-Labs) Aurélie Beynier (Lip6-CNRS) Christophe Labreuche (Thales CortAIx-Labs)
	16:35-16:55	*A queue-based approach for fine-tuning the efficiency-fairness tradeoff in distributed satellite scheduling*	Shai Krigman (Ariel University) Tal Grinshpoun (Ariel University) Lihi Dery (Ariel University)
	16:55-17:15	*Onboard Planning and Scheduling for Autonomous Satellite Operations Towards Multi-Agent Cooperation*	Riccardo Maderna (AIKO Srl) Marco Morgese (AIKO Srl)
	17:15-17:35	*Effects of Lookahead and Communication Dynamics on Multi-Agent Potential-Field Rover Navigation*	Timothy Flavin (University of Tulsa) Sandip Sen (University of Tulsa) Joe Shymanski (University of Tulsa)
17:35-17:45		**Closing remarks**	

Table of Contents

Invited Talk

Title: Developing Distributed Autonomous Space Systems

Speaker: Caleb Adams, NASA Ames Research Center

Abstract:

The development of distributed autonomous space systems is crucial for advancing NASA's Small Spacecraft and Distributed Systems (SSDS) program. The Distributed Spacecraft Autonomy (DSA) project at NASA's Ames Research Center aims to enable autonomous multi-agent decision-making in multi-spacecraft operations, mitigating the effects of communication constraints like latency and bandwidth. This talk will discuss recent advancements in DSA technologies and applications, including distributed resource and task management, reactive operations, and ad hoc network communications. In additional to technical implementations, we will discuss software development strategies for rapid on-orbit testing. We will also highlight successes from the Starling 1.0 mission, The Starling 1.5 mission extension, and a space traffic coordination technology demonstration that showcased autonomous satellite operations without human intervention. Our work focuses on building flight heritage and experience for low-TRL software, raising confidence in new autonomy and distributed control capabilities for next-generation space missions.

Speaker Biography:

Caleb Ashmore Adams is a Project Manager and Principal Investigator at the National Aeronautics and Space Administration (NASA) Ames Research Center in Silicon Valley, California. He leads the Distributed Spacecraft Autonomy (DSA) group which demonstrates advanced swarm technologies on multi-spacecraft missions. He is also the Deputy Project Manager of the Starling mission, a 4-satellite swarm. On Starling, DSA demonstrated the first fully autonomous distributed space mission, and it continues to accomplish firsts-in-space while in operations today.

Due to the success of DSA, Caleb was selected by NASA’s Space Technology Mission Directorate (STMD) to lead the preliminary design study of NASA’s Next Generation Multi-Agent Swarm. He is also the Small Business Innovative Research (SBIR) Topic Manager for Autonomous Systems under NASA’s Exploration Systems Development Mission Directorate (ESDMD) where he oversees investments into US industry that support NASA’s mission to explore our solar system.

Caleb is the Project Manager and Principal Investigator for the Low-Light Universal Mapping for Extreme Environments (LUMEN) team where he works closely with researchers across industry and academia (Stanford, UC Berkeley, Nvidia, Google) to apply modern AI and computer vision techniques to the challenging environment of space.

Caleb won an Early Career Achievement award from NASA's STMD in 2024 and was awarded NASA's STMD's highly competitive Early Career Initiative (ECI) in 2025. Under his leadership, the DSA and Starling teams have won 3 separate group achievement awards. He holds bachelor's and master's degrees in computer science from the University of Georgia, was featured as one of the University of Georgia's 40 under 40 in 2025, and a recipient of the University of Georgia's Graduate School's 2025 Alumni of Distinction Award.

Invited Talk

Title: Priority-Sensitive Mission Continuity for Degraded Multi-Orbit Satellite Constellations

Speaker: Daniel Reynolds, Massachusetts Institute of Technology

Abstract:

Satellite mega-constellations are being deployed to provide resilient global service, yet collisions, severe space weather, and degraded ground coordination threaten the legacy command-and-control assumptions on which those systems depend. This paper evaluates satellite task re-prioritization within a hybrid Mean Field Game Theory (MFGT) and Model Predictive Control (MPC) distributed fallback autonomy framework for heterogeneous multi-orbit constellations under severe degradation. The framework converts task backlog, priority concentration, agent losses, and regional service memory into a broadcast incentive heatmap that re-prioritizes task selection across the surviving constellation, while onboard receding-horizon task selection enforces local feasibility to generate optimal execution threads. In a representative stress event, a 500-satellite, 6-ground-station architecture is reduced to 146 satellites and 2 ground stations before a geographically concentrated 101-task high-priority workload is injected. The result is priority-sensitive capacity redirection under degraded control. All 101 urgent tasks are admitted immediately and completed on time. At injection, 101 of 104 commitments target the urgent workload. Across the run, the framework completes 1,508 of 1,618 tasks, yielding a task completion rate of 93.2% and a priority satisfaction index of 96.3%. These results show degraded-belief fallback autonomy can redirect post-loss capacity fast enough to preserve mission continuity under severe degradation.

Speaker Biography:

Major Daniel Reynolds is an active duty officer in the United States Space Force and a Secretary of the Air Force STEM PhD Fellow in the Department of Aeronautics and Astronautics at the Massachusetts Institute of Technology (MIT), where he is a member of the Space Telecommunications, Astronomy, and Radiation (STAR) Laboratory under the guidance of Prof. Kerri Cahoy. His research focuses on distributed autonomy for resilient space operations, with an emphasis on scalable coordination and control frameworks for heterogeneous satellite constellations operating across multiple orbits under degraded and disrupted conditions. By combining game theory, predictive control, and autonomous decision-making methods, he is working to develop architectures that enable satellites to continue making locally executable, mission-aware decisions even when centralized coordination is limited or unavailable.

Daniel has nearly a decade of experience spanning academic, nonprofit, and U.S. government-service space initiatives. He earned his bachelor's degree in Astronautical Engineering from the United States Air Force Academy in 2017 and his master's degree in Aeronautics and Astronautics from the Massachusetts Institute of Technology in 2019. In 2021, he completed the U.S. Air Force Test Pilot School's Space Test Course, which provided hands-on training in flight-test fundamentals, advanced space system testing, and space operations. Taken together, his academic, operational, and cross-sector experiences inform a research approach centered on operational realism, rigorous system-level evaluation, and the design of autonomy architectures for demanding mission environments.

Priority-Sensitive Mission Continuity for Degraded Multi-Orbit Satellite Constellations

Daniel Reynolds
Massachusetts Institute of Technology
Cambridge, United States
dcr@mit.edu

Kerri Cahoy
Massachusetts Institute of Technology
Cambridge, United States
kcahoy@mit.edu

Olivier de Weck
Massachusetts Institute of Technology
Cambridge, United States
deweck@mit.edu

ABSTRACT

Satellite mega-constellations are being deployed to provide resilient global service, yet collisions, severe space weather, and degraded ground coordination threaten the legacy command-and-control assumptions on which those systems depend. This paper evaluates satellite task re-prioritization within a hybrid Mean Field Game Theory (MFGT) and Model Predictive Control (MPC) distributed fallback autonomy framework for heterogeneous multi-orbit constellations under severe degradation. The framework converts task backlog, priority concentration, agent losses, and regional service memory into a broadcast incentive heatmap that re-prioritizes task selection across the surviving constellation, while onboard receding-horizon task selection enforces local feasibility to generate optimal execution threads. In a representative stress event, a 500-satellite, 6-ground-station architecture is reduced to 146 satellites and 2 ground stations before a geographically concentrated 101-task high-priority workload is injected. The result is priority-sensitive capacity redirection under degraded control. All 101 urgent tasks are admitted immediately and completed on time. At injection, 101 of 104 commitments target the urgent workload. Across the run, the framework completes 1,508 of 1,618 tasks, yielding a task completion rate of 93.2% and a priority satisfaction index of 96.3%. These results show degraded-belief fallback autonomy can redirect post-loss capacity fast enough to preserve mission continuity under severe degradation.

KEYWORDS

Multi-Agent Systems, Satellite Mega-Constellations, Distributed Fallback Autonomy, Mean Field Game Theory, Model Predictive Control, Degraded Operations, Priority-Sensitive Task Re-Prioritization, Resilient Space Systems

ACM Reference Format:
Daniel Reynolds, Kerri Cahoy, and Olivier de Weck. 2026. Priority-Sensitive Mission Continuity for Degraded Multi-Orbit Satellite Constellations. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 10 pages.

1 INTRODUCTION

Earth orbit is becoming a more congested, fragile, and operationally consequential domain. As of October 2025, more than 54,000 trackable objects larger than 10 cm orbit Earth alongside an estimated 1.2 million debris fragments larger than 1 cm [1]. The European Space Agency warns that, when current launch traffic, fragmentation behavior, and disposal performance are projected forward, long-term collision risk in low Earth orbit reaches roughly four times the agency's orbital sustainability threshold, a benchmark representing an acceptable risk level for a stable long-term environment [1]. The Outer Space Institute's CRASH Clock gives that risk operational immediacy, estimating on 20 March 2026 that, in a severe loss-of-control scenario, a catastrophic collision in LEO could occur within just three days [43]. Kinetic hazards are not the only system-level threat: a Carrington-class geomagnetic storm remains a statistically credible event that could degrade or disable satellites and communications infrastructure across multiple orbital regimes [2, 3]. Against this increasingly hazardous backdrop, mega-constellations of hundreds to thousands of spacecraft are rapidly becoming the dominant architectural paradigm for space-based communications services [4–6], promising persistent global coverage amidst a greater global dependence on an increasingly unpredictable space domain. The implication for mega-constellation command and control is stark: future space systems must be prepared to operate under conditions in which asset loss, stale global state, fragmented communications, and urgent regional demand are not edge cases, but expected stressors.

Constellation command and control (C2) can be viewed along a progression from centralized, to decentralized, to distributed architectures, as illustrated in Figure 1. In centralized C2, planning, scheduling, monitoring, and reassignment are concentrated in a primary ground authority, with spacecraft acting primarily as execution nodes for ground-generated commands [9, 10, 13, 46, 58]. This model provides coherent global oversight, but becomes increasingly brittle as constellation size, task tempo, contact intermittency, and disruption severity grow, because mission execution remains dependent on timely ground coordination and refreshed global state [9, 45, 58]. Decentralized C2 moves beyond this single-node model by distributing operational authority across multiple ground stations, gateways, or regional control nodes, reducing bottlenecks and better matching the parallelism of proliferated mega-constellation operations [11, 13–15, 23, 26, 36, 42, 51]. However, even decentralized ground-centric control can remain vulnerable when communication paths fragment, global state becomes stale, or terrestrial coordination layers are themselves degraded. Distributed C2 therefore represents the critical fallback autonomy step: it embeds tactical decision-making and adaptation within the constellation, allowing spacecraft to operate as autonomous agents under partial observability, intermittent communication, dynamic membership, and degraded coordination [8, 13, 23, 24, 47, 48, 56]. The unresolved issue for fallback operations is therefore not simply whether decision authority can move away from a central ground node, but whether a degraded constellation can still align local

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus.

onboard task choices with mission urgency after the coordination substrate itself has degraded.

This paper addresses that fallback-C2 problem by developing and evaluating a distributed autonomy framework for priority-sensitive task selection in a degraded heterogeneous multi-orbit constellation. The fundamental objective is to preserve mission continuity by enabling surviving satellites to recognize where urgent demand is concentrated, redirect limited capacity accordingly, and continue making feasible onboard tasking decisions when ground-mediated coordination is no longer sufficient. The paper makes two focused contributions.

Contributions: First, this work introduces a heatmap-driven distributed task-selection mechanism that converts regional backlog, agent scarcity, task priority, and recent service effectiveness into a broadcast coordination field for guiding local satellite decisions under degraded belief. Second, this work demonstrates adaptive high-priority re-prioritization after severe architectural loss, showing that a survivor constellation can redirect task commitments toward an urgent regional workload after most satellites and ground stations have been removed. Together, these contributions frame distributed fallback autonomy as a direct answer to the future operating environment of space systems, described by the ability to identify what tasks matter most, shift scarce capacity toward it, and continue executing the mission when legacy C2 assumptions fail.

2 BACKGROUND

Distributed fallback autonomy for satellite mega-constellations is not a single-property problem, but a coupled architectural challenge at the intersection of five domains: An operationally useful architecture must scale to large constellations without centralized optimization, dense pairwise coordination, or all-to-all exchange becoming the dominant bottleneck, as motivated by large-population and mean-field formulations [30, 37, 38, 41, 52]. It must support heterogeneous spacecraft, because sensing, relay, communications, storage, power, and service roles are not uniform across operational fleets [12, 21, 35, 39, 40]. It must accommodate multi-orbit structure, since LEO, MEO, and GEO assets differ in geometry, latency, persistence, coverage, and relay utility [19–21, 27, 52]. It must remain useful under degraded operations, including communication loss, partial observability, fragmentation, and stale global context [7, 31, 54–56]. Finally, it must support bounded isolated operations, allowing agents to continue locally rational behavior when timely access to global coordination signals is interrupted [17, 25, 63].

2.1 Scalability

Mean-field and aggregate-population methods provide one path to tractability by replacing multi-agent interactions with low-dimensional population descriptors or measure flows [29, 30, 37, 38, 41]. Rather than requiring each agent to reason over every other agent, these formulations couple local decisions to aggregate state, reducing the coordination burden associated with large interacting populations [18, 30, 37, 41, 52].

This scalability argument becomes operationally important after capacity loss, when the system must rapidly express where remaining capacity is most needed without solving a centralized reassignment problem over all agents and tasks. Prior work extends mean-field and aggregate coordination ideas toward engineering, partial-information, and networked-control settings [16, 31, 49, 50, 55, 56, 62, 64]. Other approaches reduce complexity through structural sparsification, tier-level control variables, local interference neighborhoods, or parallelized equilibrium search [21, 39, 60]. These results support the premise that large-population coordination can be compressed into aggregate signals, including signals that may be imperfect, delayed, or partially observed. However, aggregate coordination alone does not guarantee that a specific spacecraft can execute a specific mission task under timing, access, queue, power, storage, and communication constraints. That gap motivates the mechanism studied here: a degraded-belief regional coordination signal must be paired with local feasibility-constrained task selection.

2.2 Heterogeneity and Multi-Orbit Regions

Operational constellations are not composed of interchangeable agents. Satellites may differ in sensing roles, relay functions, communications payloads, storage capacity, power state, downlink capability, and resilience under disruption. The reviewed literature introduces heterogeneity in several forms, including heterogeneous task graphs and offloading decisions [12], multi-tier LEO network control [21], satellite/UAV-assisted edge computing [40], hybrid-constellation spectrum sharing [39], and satellite-enabled learning or selection problems [35]. Graph-based and coalition-based resource coordination likewise show that network relationships and cooperative structure matter for distributed decision-making [19, 20, 59].

Multi-orbit structure further constrains which agents can service which tasks. In the present architecture, LEO, MEO, and GEO assets play differentiated tactical, regional-relay, and global-support roles; more generally, prior work treats orbital structure through differences in geometry, visibility, routing opportunity, contact windows, and relay utility [19–21, 27, 52, 57]. These formulations show that orbital layer and network geometry shape coordination opportunity. A degraded survivor constellation may retain some regions, paths, or task classes better than others, so post-loss capacity is not merely reduced; it is redistributed unevenly across altitude, coverage, relay access, and service opportunity.

2.3 Degraded and Isolated Operations

Distributed autonomy is most valuable precisely when the coordination substrate is damaged, delayed, or incomplete. The reviewed literature includes several ingredients relevant to this regime: adversarial or unreliable topology in dynamic connectivity games [7], spoofing or corrupted topology discovery in graph-based satellite coordination [19, 20], intermittent contact and storage-limited forwarding in time-expanded routing [27], peer relay and caching under poor channel conditions [33], imperfect monitoring in repeated games [54], and partial-observation or disturbance-rejection treatments in mean-field settings [31, 55, 56]. Local feasibility under communication limits is also supported in decentralized navigation and MPC-oriented settings [17, 25, 61].

These works provide important degraded-operation ingredients, but the present setting emphasizes a more specific operational mode: the constellation has already lost substantial capacity, the

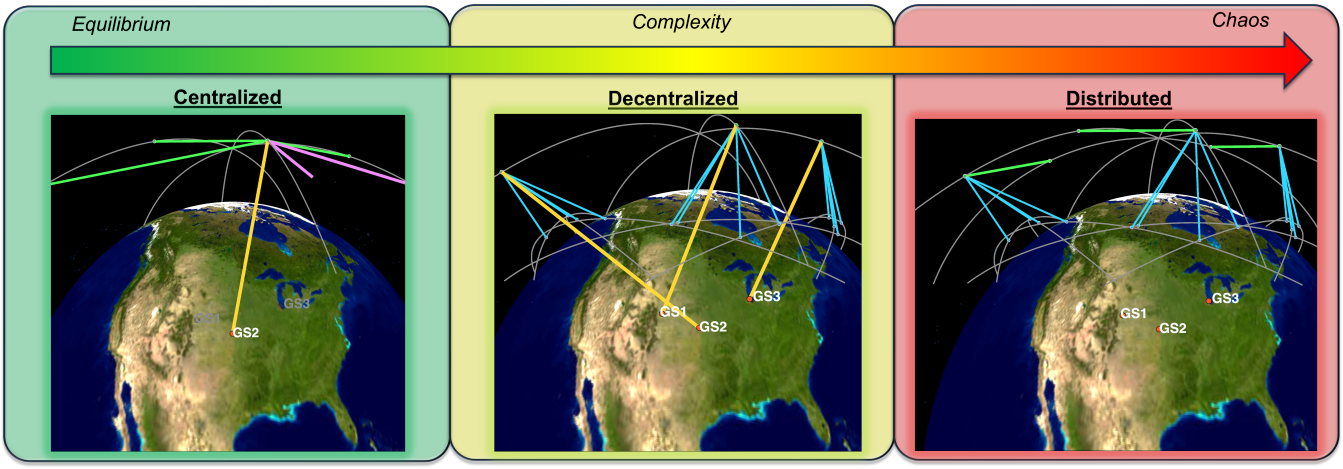


Figure 1: Progression of satellite coordination architectures from centralized to decentralized and distributed control.

ground segment is degraded, and urgent regional work appears after the disruption. Under those conditions, the coordination signal may be stale, delayed, fragmented, or available only through intermittent refresh. Many distributed approaches still depend on iterative exchange, negotiation, reward feedback, neighbor updates, or reliable broadcast information to preserve coordination quality [22, 35, 39, 49, 50, 53, 57, 63]. The unresolved issue is therefore not simply robustness to noisy information. It is whether degraded agents can still redirect scarce capacity toward urgent mission demand when the information needed for coordination is itself imperfect.

Isolation is the limiting case of degraded command and control: an agent may lose timely access to the current constellation state, may be unable to refresh the coordination field, and may still need to choose a feasible action under mission, timing, power, storage, and communication constraints. Prior work provides examples of communication-limited local control, bounded-neighbor reasoning, decentralized response under imperfect monitoring, and transitions away from hub-mediated coordination after link degradation [17, 25, 54, 63]. These works establish that meaningful local behavior can persist when global exchange weakens.

3 FALLBACK AUTONOMY FRAMEWORK

The framework developed in this paper exists at the intersection of the five operational domains identified above through a hybrid Mean Field Game Theory (MFGT) and Model Predictive Control (MPC) architecture. MFGT addresses scalability by compressing large-population interaction into a regional tasking heatmap, while the heatmap-based representation captures demand, scarcity, priority, and recent service history across a heterogeneous multi-orbit operating area. MPC addresses local executability by converting that coordination pressure into onboard task-selection decisions that respect orbit-dependent geometry, timing, queueing, power, storage, and connectivity constraints. Under degraded or partially isolated conditions, satellites act from perceived or cached coordination context rather than relying on centralized reassignment or dense pairwise negotiation. Together, the two layers form a distributed fallback autonomy architecture: a low-bandwidth coordination field identifies where scarce capacity should flow, while each satellite independently determines what it can actually execute.

Mean Field Game Theory (MFGT) provides a scalable way to reason about decision-making in very large multi-agent systems. Rather than modeling every pairwise interaction among agents, mean-field methods approximate the collective influence of the population through an aggregate state, distribution, or measure flow; each agent then computes its response to that aggregate population behavior rather than to every other individual agent [29, 30, 37, 38, 41]. This abstraction preserves the core strategic coupling of a large decision population while avoiding the coordination burden that would arise from dense all-to-all interaction [18, 30, 37, 41, 52]. That structure is well matched to degraded satellite mega-constellation operations: the constellation is large, coupling is driven by regional demand and aggregate agent scarcity, and centralized full-state optimization becomes progressively less tenable as the system scales or degrades [16, 31, 49, 50, 55, 56, 62, 64]. In this work, the mean-field layer implements that idea as a regional broadcast signal: task backlog, agent scarcity, task priority, and recent service effectiveness are compressed into a low-dimensional heatmap that guides local satellite decisions without centralized reassignment, persistent all-to-all coordination, or dense pairwise negotiation.

MFGT alone, however, does not determine whether a specific satellite can physically execute a specific task under degraded conditions. Prior mean-field work generally emphasizes aggregate coordination more directly than task-level execution feasibility, leaving timing, storage, power, queueing, and serviceability constraints to be handled by an additional local control layer [28, 31, 49, 50, 64]. Model Predictive Control (MPC) provides that execution layer. MPC is a receding-horizon control framework in which an agent repeatedly solves a finite-horizon decision problem from its current state, evaluates the predicted consequences of candidate actions, applies only the first selected action, and then replans when the next state is observed [32, 34, 44, 61]. This structure is useful for fallback

autonomy because onboard tasking is not a one-time assignment problem: each satellite must continually decide whether to idle, continue current work, or commit to a new task as orbital access, task deadlines, battery state, storage state, queue load, connectivity, and degraded information evolve [17, 25, 61]. In this work, the MPC layer converts the mean-field heatmap into executable local task-selection decisions by filtering infeasible tasks, scoring admissible actions over a finite horizon, executing the first action, and repeating the process at the next decision epoch. Prior work has shown that mean-field coordination and receding-horizon control can be interfaced in a meaningful way, although not for the full heterogeneous, multi-orbit, degraded-operations constellation setting addressed here [32, 34, 44, 61].

3.1 Mean Field Game Theory

The executable MFGT layer is implemented as a discrete, tiled, simulation-first coordination mechanism rather than as a continuous PDE solver. At each decision epoch, task demand and satellite availability are aggregated over geographic tiles. Each region r is represented by a four-channel broadcast state,

$$\mu_r(t) = \begin{bmatrix} \rho_r(t) \\ \xi_r(t) \\ \delta_r(t) \\ \psi_r(t) \end{bmatrix}, \tag{1}$$

where $\rho_r(t)$ is backlog density, $\xi_r(t)$ is agent scarcity, $\delta_r(t)$ is priority concentration, and $\psi_r(t)$ is service-effectiveness memory. These four terms are designed to capture the mission-relevant conditions that should attract limited surviving capacity during degraded operations.

Backlog density is computed from the number of outstanding, not-completed tasks in region r, normalized by the relative tile-area scale,

$$\rho_r(t) = \frac{|\mathcal{T}_r^{\text{out}}(t)|}{\alpha_r}. \tag{2}$$

Agent scarcity is computed from the number of active satellites that can currently service the region,

$$n_r(t) = \sum_{i \in \mathcal{A}(t)} \mathbf{1}\{i \rightsquigarrow r \text{ at } t\}, \qquad \xi_r(t) = \text{clip}_{[0,1]}\left(\frac{\eta^{\max} - n_r(t)/\alpha_r}{\eta^{\max}}\right). \tag{3}$$

Priority concentration is the mean normalized priority of outstanding tasks in the region,

$$\delta_r(t) = \begin{cases} \dfrac{1}{|\mathcal{T}_r^{\text{out}}(t)|} \displaystyle\sum_{\tau \in \mathcal{T}_r^{\text{out}}(t)} \pi_\tau(t), & |\mathcal{T}_r^{\text{out}}(t)| > 0, \\ 0, & |\mathcal{T}_r^{\text{out}}(t)| = 0, \end{cases} \tag{4}$$

where $\pi_\tau(t) \in [0, 1]$. Finally, service-effectiveness memory is updated as an exponential moving average,

$$\psi_r(t) = (1 - \beta_\psi)\, \psi_r(t - \Delta t) + \beta_\psi\, s_r(t - \Delta t), \tag{5}$$

where $s_r(t - \Delta t)$ is the recent regional completion-fraction proxy. This term prevents the broadcast field from reacting only to instantaneous backlog and allows recent service history to influence regional incentives.

The scalar broadcast incentive for each region is then

$$\Phi_r(t) = \gamma^\top \mu_r(t), \tag{6}$$

where $\gamma \in \mathbb{R}^4$ is a fixed weight vector, meaning that the coordination field is driven entirely by the observed regional mission state. Operationally, $\Phi_r(t)$ functions as a tasking heatmap: regions with high unmet demand, limited available service, high-priority tasks, or poor recent service exert stronger attraction on the surviving constellation.

Because degraded operations may interrupt access to the current broadcast field, each satellite acts on a perceived incentive rather than the true global field. Satellite i maintains a local estimate $\hat{\mu}_{i,r}(t)$, refreshed when communications permit and otherwise retained from cached information. Its perceived regional incentive is

$$\hat{\Phi}_{i,r}(t) = \gamma^\top \mathcal{T}\big(\hat{\mu}_{i,r}(t)\big) + \lambda_r(t), \tag{7}$$

where $\mathcal{T}(\cdot)$ denotes the policy-facing feature transform used by the implementation. This perceived incentive is the only population-level coordination signal used by the local controller. Physical feasibility remains enforced onboard.

3.2 Model Predictive Control

The MPC layer converts the perceived regional incentive field into executable local task-selection decisions. At each epoch, satellite i observes its local state, including orbital access, battery, storage, queue load, resilience state, and connectivity posture. It then constructs a locally feasible candidate task set,

$$\begin{aligned} \mathcal{K}_i^{\text{feas}}(t) = \Big\{ k \in \mathcal{K}(t) \;:\; & \text{svc}_i(k, t) = 1, \\ & t \geq t_k^{\min}, \\ & t + d_k(t) \leq t_k^{\max}, \\ & x_i^{\text{queue}}(t) < Q_i^{\max}, \\ & x_i^{\text{bat}}(t) \geq e_k, \\ & x_i^{\text{stor}}(t) + m_k \leq S_i^{\max} \Big\}. \end{aligned} \tag{8}$$

This set encodes the core execution constraints: serviceability, timing, queue capacity, energy, and storage. Additional implementation guards enforce task-consistency, same-tick claim resolution, and degraded-mode resilience checks.

The admissible action set is composed of idle, continue, and commit actions,

$$\mathcal{U}_i(t) = \{\text{idle}\} \cup \{\text{continue current task}\} \cup \{\text{commit to } k \in \mathcal{K}_i^{\text{feas}}(t)\}. \tag{9}$$

The continue action is available when the satellite has an active queue-head task, while idle remains admissible as a fallback when new commitments are infeasible or unattractive.

For feasible commit actions, the local planner evaluates a mission-structured cost that first scores individual task commitments and then maps those scores into the finite-horizon rollout objective. The resulting stage-cost terms balance resource burden, urgency, latency, regional congestion, capability alignment, and task priority. The mean-field incentive enters this cost through the perceived regional field $\hat{\Phi}_{i,r_k}(t)$, so that tasks located in regions with stronger coordination pressure become more attractive to the onboard planner. Thus, the broadcast heatmap does not assign tasks directly; it biases the local finite-horizon decision problem.

Each satellite then solves a discrete receding-horizon problem,

$$\pi_i^\star(t) \in \arg\min_{\pi \in \Pi_i(t)} \sum_{h=0}^{H-1} \ell_i\Big(\tilde{x}_i(t+h\Delta t),\, \pi(h),\, \hat{\Phi}_{i,\cdot}(t+h\Delta t)\Big), \quad (10)$$

where $\Pi_i(t)$ is the set of admissible horizon-length action sequences, $\tilde{x}_i(\cdot)$ is the rollout state under the planner's internal progression model, and $\ell_i(\cdot)$ is the mission-structured stage cost for idle, continue, or commit actions. In the implemented controller, this stage cost is parameterized by tuned weights on resource preservation, urgency, latency, congestion, alignment, and priority. Only the first action is applied,

$$u_i^\star(t) = \pi_i^\star(t; 0), \quad (11)$$

and the optimization is repeated at the next decision epoch.

This receding-horizon structure allows each satellite to respond to new tasks, changing resource states, evolving orbital access, and degraded broadcast information without requiring centralized reassignment. The implemented controller is therefore not a generic quadratic MPC regulator; it is a mission-structured finite-horizon task planner. The MFGT layer supplies regional coordination pressure through $\hat{\Phi}_{i,r}(t)$, while the MPC layer determines whether acting on that pressure is locally feasible and mission-beneficial.

4 RESULTS AND ANALYSIS

The simulation spans 288 one-minute decision epochs, or approximately 4.8 hours, over a 15×10 southeastern United States (SEUS) regional grid. The initial architecture contains a heterogeneous 500-satellite constellation composed of 400 LEO, 96 MEO, and 4 GEO satellites, supported by six ground stations and initialized with 656 tasks split evenly between remote-sensing and communications demand. During nominal operations, three new tasks are injected per epoch, corresponding to one task-injection opportunity every 60 seconds. At step 96, or t = 1:36:00, a solar-storm-driven mass-casualty event removes 354 satellites, approximately 71% of the constellation, and four ground stations, approximately 67% of the ground segment. The surviving architecture therefore contains 146 satellites and two active ground stations, with the remaining space segment composed of 120 LEO, 24 MEO, and 2 GEO satellites. Eleven minutes later, at step 107, or t = 1:47:00, a geographically concentrated hurricane-response workload injects 101 high-priority tasks across eight coastal tiles in the Florida Atlantic corridor. This event sequence creates a compound stress test: severe architecture loss first reduces available capacity and ground access, then urgent regional demand appears after the constellation is already operating in a degraded state. All controller settings are held fixed across the run, so the observed response reflects closed-loop adaptation rather than phase-specific retuning. The broadcast field is weighted to emphasize priority concentration, backlog pressure, and agent scarcity, while retaining a smaller service-memory term to smooth short-horizon regional response. The onboard receding-horizon planner is tuned to favor urgent and high-priority commitments while still penalizing resource burden, latency, congestion, and poor capability alignment.

4.1 Heatmap-Driven Distributed Task Selection

Figure 3 shows the mechanism behind the first contribution: task selection is driven by a regional incentive field rather than by a centralized reassignment list. The MFGT layer compresses outstanding task density, agent scarcity, priority concentration, and recent service memory into the scalar field $\Phi_r(t)$. Each satellite then sees that field through its local, possibly stale estimate and passes the resulting regional incentive into its onboard receding-horizon task planner. The heatmap effectively serves as the coordination object that changes which feasible commitments look mission-beneficial to locally planning satellites.

The four event-aligned stills show the field responding to the actual mission state. At the initial epoch, the field is broad because the seeded workload is large and widely distributed. By the pre-constellation loss frame, the nominal controller has mostly absorbed that initial backlog, and the field is correspondingly lower and more localized. The step-96 constellation loss changes the interpretation of the remaining work: even with little pending demand, the loss of servicing capacity and ground infrastructure makes regional scarcity more consequential. At step 107, the high-priority injection creates a concentrated ridge over the Florida Atlantic corridor, making the urgent region the dominant attractor for survivor capacity.

The trace metrics show that this visual field shift is paired with a real change in task-selection behavior. The architecture does not preserve nominal volume after the loss event: completed-task flow falls from $860/96 = 8.96$ tasks per step in the nominal phase to $648/192 = 3.38$ tasks per step under constellation loss, and accepted commitments fall from 9.7812 to 3.9427 per step. Queue load contracts as well, with mean queued active pairs dropping from 220.49 before loss to 100.71 afterward. The test therefore exposes the controller to a genuine reduction in available execution capacity before the urgent workload appears.

Within that lower-capacity regime, however, the value signal is preserved more strongly than the raw task count. Priority-weighted completion throughput decreases from 0.9086 in the nominal phase to 0.7944 after loss, then rises to 0.8257 during the high-priority phase. This pattern is the central evidence for heatmap-driven task selection: the post-loss constellation cannot recover nominal task volume, but the broadcast field redirects the remaining feasible commitments toward higher-value demand. At the injection epoch, 101 of 104 accepted commitments are high-priority tasks, showing that the field concentration over the corridor is converted into local task commitments rather than remaining only a visualization artifact.

4.2 Adaptive High-Priority Re-Prioritization After Severe Loss

Figure 4 shows the closed-loop task-flow consequence of the heatmap shift. The created, assigned, and completed traces first show nominal absorption of the seeded workload. At step 96, the constellation is reduced from 500 to 146 satellites and from six to two active ground stations, producing a visible assignment response as the survivor architecture re-forms its queues. Eleven minutes later, the step-107 high-priority injection produces a much sharper selection

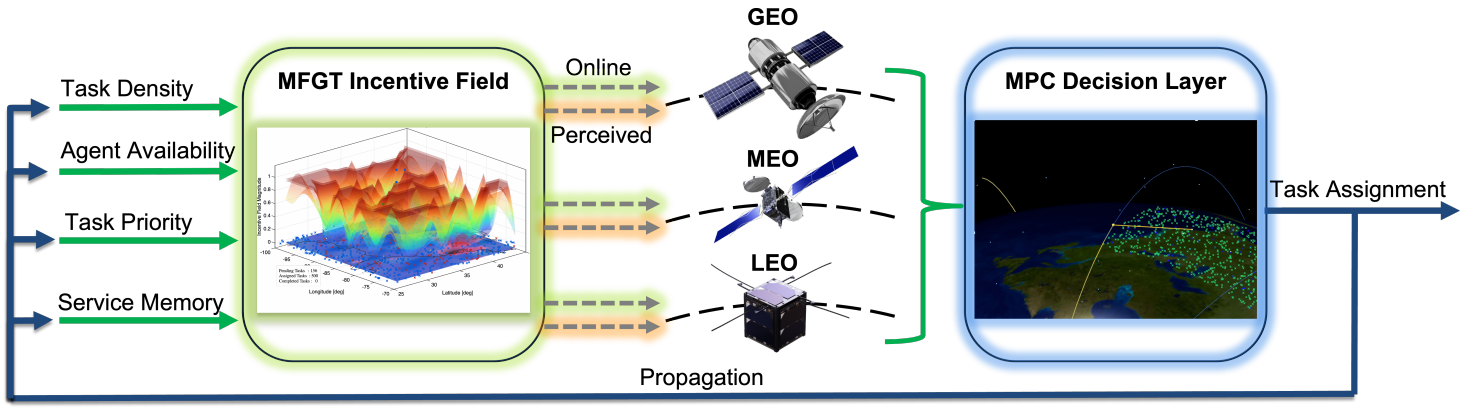


Figure 2: Framework overview showing how regional demand, availability, priority, and service memory form an MFGT incentive field that guides multi-orbit MPC task assignment.

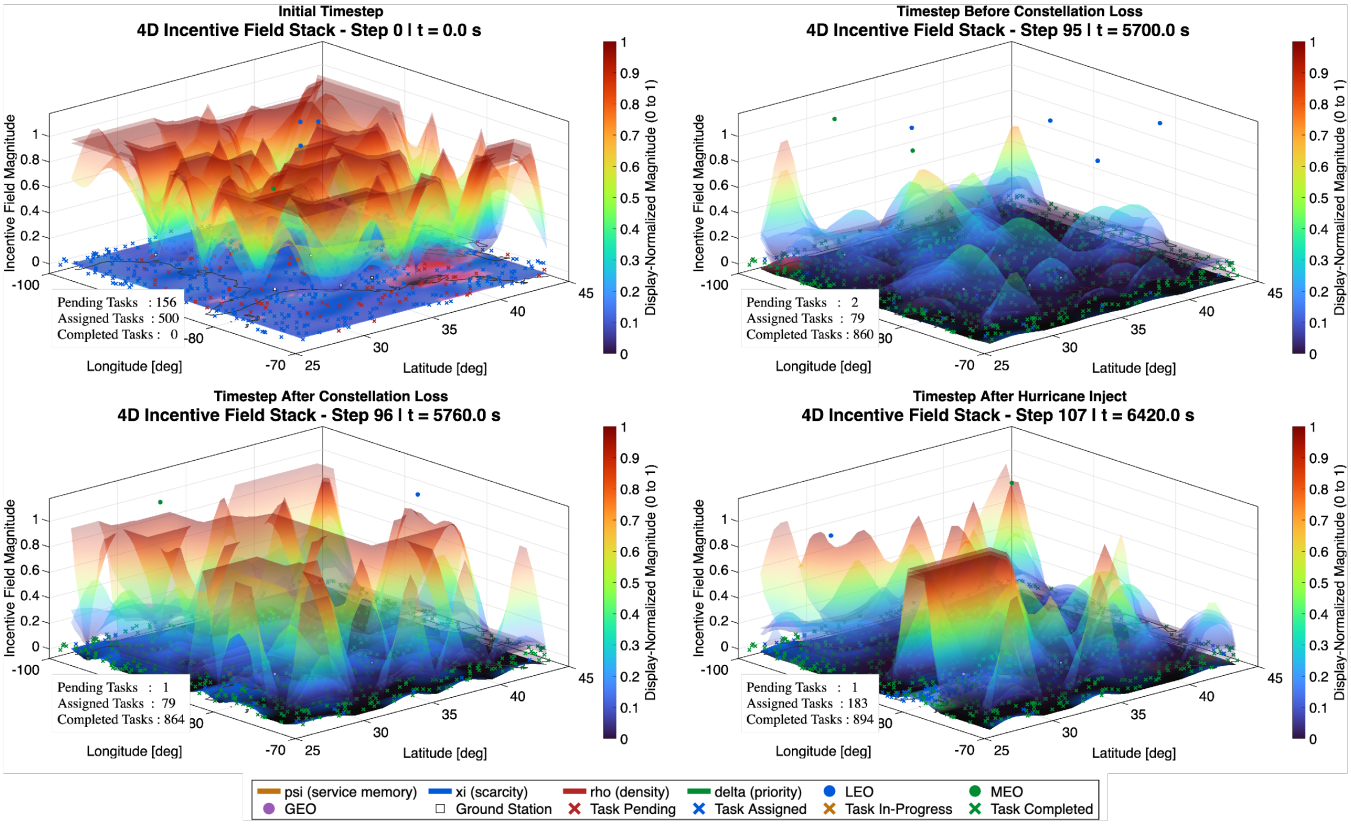


Figure 3: Event-aligned incentive-field stills showing the evolution of regional coordination pressure before contraction, after contraction, and after high-priority task injection.

event: 104 commitments are accepted at that epoch, and 101 of them are directed to the injected urgent tasks.

The high-priority response is immediate at assignment time and bounded at completion time. All 101 injected tasks are assigned at step 107, giving zero-step assignment latency at the mean, median, and 95th percentile. All 101 are also completed on time. Completion latency is tightly concentrated, with mean, median, and 95th-percentile values of 13.60, 14.00, and 15.00 steps, corresponding to approximately 816, 840, and 900 seconds at the one-minute decision cadence. The completion wave visible after the injection

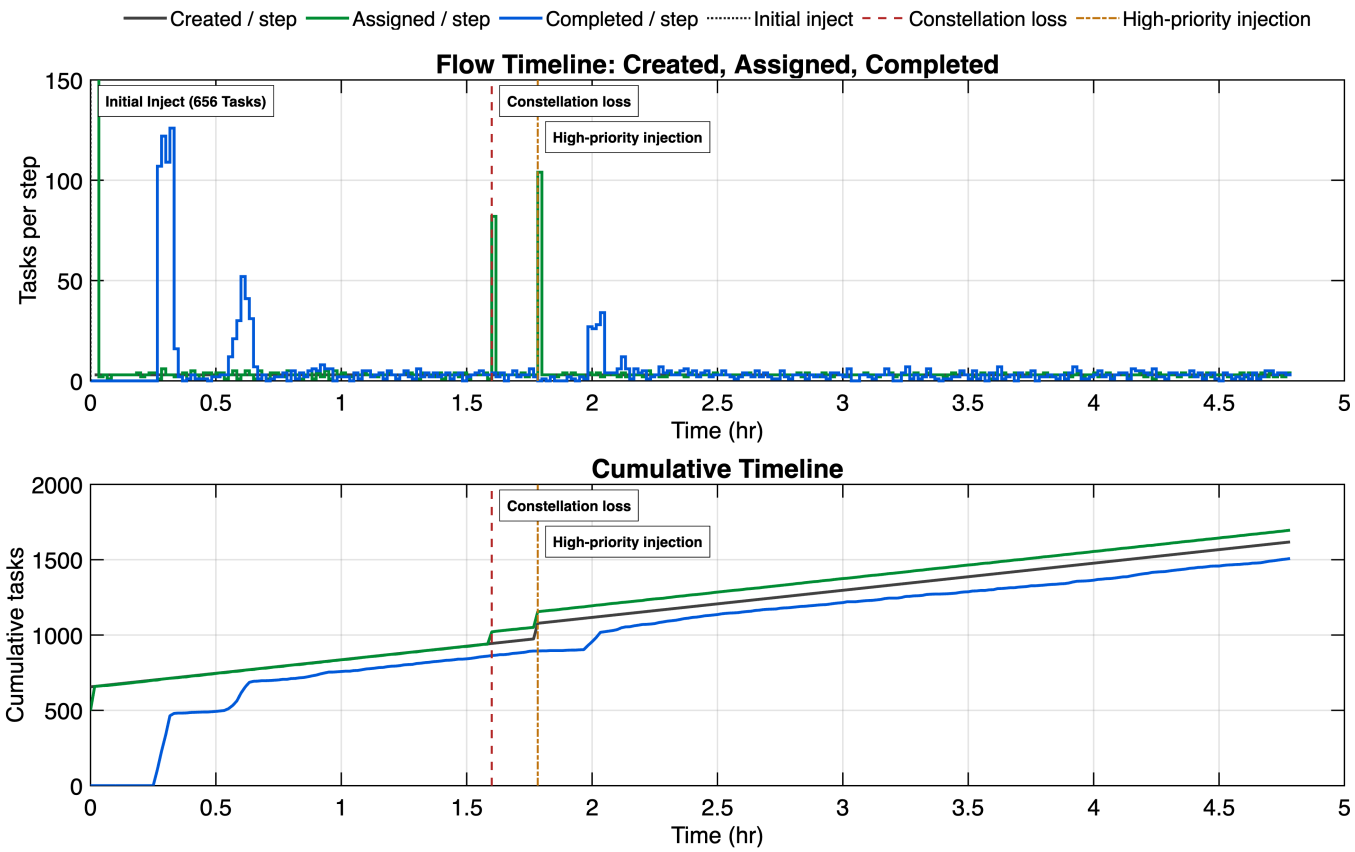


Figure 4: Created, assigned, and completed task flow and cumulative stock over the degraded SEUS run. Vertical event markers indicate the initial workload insertion, the step-96 constellation loss, and the step-107 high-priority injection.

marker is therefore the execution-time consequence of immediate re-prioritization, not delayed admission.

The aggregate mission accounting reinforces this interpretation. Across the full horizon, the framework completes 1,508 of 1,618 created tasks, yielding a 93.2% task completion rate. The priority satisfaction index is higher, at 96.3%, which means the surviving constellation preserves mission value better than it preserves raw task volume. At the final step, no tasks remain pending; the 110-task created-completed gap is already absorbed as 14 assigned tasks and 96 tasks in execution. The residual gap is therefore finite-horizon work in progress rather than unclaimed demand.

Table 1 reports the same result in compact form. The high-priority injection is fully admitted and fully completed, with an HPI on-time completion rate of 1.0000 and missed-deadline rate of 0.0000. The admission audit finds no unassigned HPI tasks, no never-feasible HPI tasks, and no serviceable-but-unselected HPI tasks. The re-prioritization claim is therefore stronger than a throughput claim: after severe loss, every injected urgent task is both selected and completed before deadline.

Replanning remains bounded even though the two disruption epochs produce sharp queue updates. Queue churn peaks at 0.9921 at the step-96 loss event and reaches 0.8739 at the step-107 injection, but the run mean is only 0.0997. Natural queue removals account for

Table 1: Primary mission and high-priority injection outcomes for the representative degraded SEUS run.

Metric	*Value*
Mission-level performance	
Created / completed tasks	1618 / 1508
Task completion rate	0.9320
Priority satisfaction index	0.9632
On-time completion rate / missed-deadline rate	0.9320 / 0.0000
Weighted on-time completion / missed-deadline rate	0.9632 / 0.0000
High-priority injection response	
Injected high-priority tasks	101
Assigned / completed high-priority tasks	101 / 101
Assignment latency mean / p50 / p95	0/0/0 steps
Completion latency mean / p50 / p95	13.60/14.00/15.00 steps
High-priority on-time completion / missed-deadline rate	1.0000 / 0.0000
Accepted commitments at injection	104
High-priority commitments at injection	101
Replanning stability	
Queue churn mean / peak	0.0997 / 0.9921
Final pending / assigned / in-execution	0 / 14 / 96

1,504 completed-task departures, while decision-driven removals sum to 78 and decision-driven reorders are zero. The controller therefore reacts strongly at the moments where the mission state actually changes, but it does not exhibit persistent queue thrashing. Taken together, the heatmap and timeline show adaptive high-priority re-prioritization rather than nominal-service preservation: the degraded constellation loses volume, shifts coordination pressure toward the Florida Atlantic corridor, and spends its remaining feasible capacity on the mission-critical workload.

5 CONCLUSION

The SEUS test campaign demonstrated priority-sensitive mission continuity for degraded heterogeneous multi-orbit satellite constellations using a hybrid MFGT–MPC distributed fallback autonomy framework. In the representative stress test, a 500-satellite, 6-ground-station architecture was reduced to 146 satellites and 2 ground stations before 101 high-priority hurricane-response tasks were injected across a concentrated Florida Atlantic corridor. The result was a direct test of whether a damaged constellation could recognize urgent demand, redirect scarce capacity, and continue executing the mission without centralized reassignment or phase-specific retuning.

The framework preserved mission value under severe loss. The heatmap-driven MFGT layer converted backlog, agent scarcity, priority concentration, and service memory into regional coordination pressure, while the MPC layer translated that pressure into feasible onboard commitments. At the injection epoch, 101 of 104 accepted commitments targeted the urgent workload. All 101 high-priority tasks were assigned immediately, completed before deadline, and incurred zero missed deadlines. Across the full run, the framework completed 1,508 of 1,618 tasks, achieving a 93.2% task completion rate and a 96.3% priority satisfaction index.

When constellation capacity collapses and ground coordination is degraded, mission continuity depends on the surviving assets' ability to make timely, priority-aware decisions from imperfect information. Heatmap-led MFGT coordination coupled with MPC-based local feasibility provides an executable mechanism for off-nominal stress tests that are becoming much more likely in today's space domain. The distributed autonomy framework does not merely preserve task volume; it preserves mission value by directing remaining capacity toward the work that matters most.

ACKNOWLEDGMENTS

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Space Force, the Department of War, or the U.S. Government.

The authors gratefully express their sincere appreciation for the full financial support through the U.S. Department of War, the U.S. Space Force, the Air Force Institute of Technology, and the Secretary of the Air Force's Science, Technology, Engineering, and Mathematics PhD Fellowship. Generative AI was used only for limited grammatical editorial checks during preparation of the manuscript.

REFERENCES

[1] 2025. *ESA'S Annual Space Environment Report.* Technical Report. European Space Agency, Darmstadt.

[2] 2025. Flying through the biggest solar storm ever recorded. https://www.esa.int/Space_Safety/Space_weather/Flying_through_the_biggest_solar_storm_ever_recorded

[3] 2025. What a Solar Superstorm Could Mean for the US. https://www.usgs.gov/news/featured-story/what-a-solar-superstorm-could-mean-us

[4] 2026. Amazon Leo. https://www.aboutamazon.com/what-we-do/devices-services/amazon-leo

[5] 2026. *Authorization and Order: Space Exploration Holdings, LLC Request for Deployment and Operating Authority for the SpaceX Gen2 NGSO Satellite System.* Technical Report. Federal Communications Commission, Washington, D.C. 1–34 pages. https://docs.fcc.gov/public/attachments/DA-26-36A1.pdf

[6] 2026. Starlink: Satellite Technology. https://starlink.com/technology?srsltid=AfmBOorqRrP7e_UOPGvyNAF0zRHhYaqN1dIRzdKQ4PXtxyxH8a0d6qUr

[7] Nof Abuzainab and Walid Saad. 2018. Dynamic Connectivity Game for Adversarial Internet of Battlefield Things Systems. *IEEE Internet of Things Journal* 5, 1 (2 2018), 378–390. https://doi.org/10.1109/JIOT.2017.2786546

[8] Carles Araguz, Elisenda Bou-Balust, and Eduard Alarcón. 2018. Applying autonomy to distributed satellite systems: Trends, challenges, and future prospects. *Systems Engineering* 21, 5 (9 2018), 401–416. https://doi.org/10.1002/sys.21428

[9] Mohamed Khalil Ben-Larbi, Kattia Flores Pozo, Mirue Choi, Tom Haylok, Benjamin Grzesik, Andreas Haas, Dominik Krupke, Harald Konstanski, Volker Schaus, Sándor P. Fekete, Christian Schurig, and Enrico Stoll. 2021. Towards the automated operations of large distributed satellite systems. Part 2: Classifications and tools. *Advances in Space Research* 67, 11 (6 2021), 3620–3637. https://doi.org/10.1016/j.asr.2020.08.018

[10] Mohamed Khalil Ben-Larbi, Kattia Flores Pozo, Tom Haylok, Mirue Choi, Benjamin Grzesik, Andreas Haas, Dominik Krupke, Harald Konstanski, Volker Schaus, Sándor P. Fekete, Christian Schurig, and Enrico Stoll. 2021. Towards the automated operations of large distributed satellite systems. Part 1: Review and paradigm shifts. *Advances in Space Research* 67, 11 (6 2021), 3598–3619. https://doi.org/10.1016/j.asr.2020.08.009

[11] Tom Butash, Peter Garland, and Barry Evans. 2021. Non-geostationary satellite orbit communications satellite constellations history. *International Journal of Satellite Communications and Networking* 39, 1 (1 2021), 1–5. https://doi.org/10.1002/sat.1375

[12] Furong Chai, Qi Zhang, Haipeng Yao, Xiangjun Xin, Ran Gao, and Mohsen Guizani. 2023. Joint Multi-Task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT. *IEEE Transactions on Vehicular Technology* 72, 6 (6 2023), 7783–7795. https://doi.org/10.1109/TVT.2023.3238771

[13] Giacomo Curzi, Dario Modenini, and Paolo Tortora. 2020. Large Constellations of Small Satellites: A Survey of Near Future Challenges and Missions. *Aerospace* 7, 9 (9 2020), 1–18. https://doi.org/10.3390/AEROSPACE7090133

[14] Inigo del Portillo, Bruce G. Cameron, and Edward F. Crawley. 2019. A technical comparison of three low earth orbit satellite constellation systems to provide global broadband. *Acta Astronautica* 159 (6 2019), 123–135. https://doi.org/10.1016/j.actaastro.2019.03.040

[15] Inigo del Portillo, Sydney I. Dolan, Bruce G. Cameron, and Edward F. Crawley. 2023. Architectural Decisions for Communications Satellite Constellations to Maintain Profitability While Serving Uncovered and Underserved Communities. *International Journal of Satellite Communications and Networking* 41, 1 (1 2023), 82–97. https://doi.org/10.1002/sat.1464

[16] Boualem Djehiche, Alain Tcheukam, and Hamidou Tembine. 2017. Mean-Field-Type Games in Engineering. *AIMS Electronic Engineering* 1, 1 (11 2017), 18–73. https://doi.org/10.3934/ms.2017.1.18

[17] J. Mikael Eklund, Jonathan Sprinkle, and S. Shankar Sastry. 2012. Switched and symmetric pursuit/evasion games using online model predictive control with application to autonomous aircraft. *IEEE Transactions on Control Systems Technology* 20, 3 (5 2012), 604–620. https://doi.org/10.1109/TCST.2011.2136435

[18] Karthik Elamvazhuthi and Spring Berman. 2019. Mean-Field Models in Swarm Robotics: A Survey. *Bioinspiration & Biomimetics* 15, 1 (11 2019), 1–15. https://doi.org/10.1088/1748-3190/ab49a4

[19] Huilong Fan, Chongxiang Sun, Jun Long, Ling Li, Yakun Huo, and Shangpeng Wang. 2025. Graph-Driven Resource Allocation Strategies in Satellite IoT: A Cooperative Game-Theoretic Approach. *IEEE Internet of Things Journal* 12, 4 (2025), 3463–3481. https://doi.org/10.1109/JIOT.2024.3407123

[20] Xiying Fan, Di Liu, Mengxuan Qiu, Yingqi Li, Jiahao Huo, Haojin Li, and Chen Sun. 2025. Dynamic Prioritized Data Transmission Through Intersatellite Cooperation in LEO Constellations. *IEEE Internet of Things Journal* 12, 2 (2025), 1922–1932. https://doi.org/10.1109/JIOT.2024.3464531

[21] Shaohan Feng, Xiao Lu, Sumei Sun, Ekram Hossain, Guiyi Wei, and Zhengwei Ni. 2024. Covert Communication in Large-Scale Multi-Tier LEO Satellite Networks. *IEEE Transactions on Mobile Computing* 23, 12 (2024), 11576–11587. https://doi.org/10.1109/TMC.2024.3396793

[22] Yufang Gao, Zhi Ji, Kanglian Zhao, Tomaso De Cola, and Wenfeng Li. 2024. Game-Based Computation Offloading and Power Allocation for LEO Constellation Networks in Distributed and Dynamic Environment. *IEEE Internet of Things Journal* 11, 4 (2 2024), 7040–7058. https://doi.org/10.1109/JIOT.2023.3314650

[23] Nathaniel G. Gordon, Nesrine Benchoubane, Gunes Karabulut Kurt, and Gregory Falco. 2025. On the Role of Communications for Space Domain Awareness. *Journal of Aerospace Information Systems* 22 (12 2025), 1–12. https://doi.org/10.2514/1.i011629

[24] Mohamad A. Hady, Siyi Hu, Mahardhika Pratama, Jimmy Cao, and Ryszard Kowalczyk. 2025. Multi-Agent Reinforcement Learning for Autonomous Multi-Satellite Earth Observation: A Realistic Case Study. (11 2025). http://arxiv.org/abs/2506.15207

[25] B Haluk and H Bozma. 2010. Multi-robot Navigation with Limited Communication - Deterministic vs Game-Theoretic Networks. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Taipei, 1825–1830.

[26] Joshua Holder, Spencer Kraisler, and Mehran Mesbahi. 2025. Centralized and Distributed Strategies for Handover-Aware Task Allocation in Satellite Constellations. *Journal of Guidance, Control, and Dynamics* 48, 6 (6 2025), 1201–1210. https://doi.org/10.2514/1.G008363

[27] Ye Hu, Walid Saad, T Charles Clancy, Jeffrey H Reed, Kekatos Vassilis, and Craig A Woolsey. 2021. *Game Theory and Meta Learning for Optimization of Integrated Satellite-Drone-Terrestrial-Communication Systems*. Ph.D. Dissertation. Virginia Polytechnic Institute and State University, Blacksburg.

[28] Han Huang, Jiajia Yu, Jie Chen, and Rongjie Lai. 2023. Bridging mean-field games and normalizing flows with trajectory regularization. *J. Comput. Phys.* 487 (8 2023). https://doi.org/10.1016/j.jcp.2023.112155

[29] Kuang Huang, Xuan Di, Qiang Du, and Xi Chen. 2020. A Game-Theoretic Framework for Autonomous Vehicles Velocity Control: Bridging Microscopic Differential Games and Macroscopic Mean Field Games. (12 2020). https://doi.org/10.3934/dcdsb.2020131

[30] Minyi Huang, Roland Malhame, and Peter Caines. 2006. Large Population Stochastic Dynamic Games: Closed-Loop McKean-Vlasov Systems and the Nash Certainty Equivalence Principle. *Communications in Information and Systems* 6, 3 (2006), 221–252.

[31] Pengyan Huang, Guangchen Wang, Shujun Wang, and Hua Xiao. 2024. A Mean-Field Game for a Forward-Backward Stochastic System with Partial Observation and Common Noise. *IEEE/CAA Journal of Automatica Sinica* 11, 3 (3 2024), 746–759. https://doi.org/10.1109/JAS.2023.124047

[32] Daisuke Inoue, Yuji Ito, Takahito Kashiwabara, Norikazu Saito, and Hiroaki Yoshida. 2021. Model Predictive Mean Field Games for Controlling Multi-Agent Systems. *arXiv* (8 2021), 1–10. http://arxiv.org/abs/2004.07994

[33] Chunxiao Jiang and Zhen Li. 2020. Decreasing Big Data Application Latency in Satellite Link by Caching and Peer Selection. *IEEE Transactions on Network Science and Engineering* 7, 4 (10 2020), 2555–2565. https://doi.org/10.1109/TNSE.2020.2994638

[34] Julian Barreiro-Gomez, Tyrone E. Duncan, and Hamidou Tembine. 2019. Linear-Quadratic Mean-Field-Type Games-Based Stochastic Model Predictive Control: A Microgrid Energy Storage Application. In *2019 American Control Conference (ACC)*. AACC, Philadelphia, 3224–3229.

[35] Yuhan Kang, Yifei Zhu, Dan Wang, Zhu Han, and Tamer Basar. 2024. Joint Server Selection and Handover Design for Satellite-Based Federated Learning Using Mean-Field Evolutionary Approach. *IEEE Transactions on Network Science and Engineering* 11, 2 (3 2024), 1655–1667. https://doi.org/10.1109/TNSE.2023.3328776

[36] Oltjon Kodheli, Eva Lagunas, Nicola Maturo, Shree Krishna Sharma, Bhavani Shankar, Jesus Fabian Mendoza Montoya, Juan Carlos Merlano Duncan, Danilo Spano, Symeon Chatzinotas, Steven Kisseleff, Jorge Querol, Lei Lei, Thang X. Vu, and George Goussetis. 2021. Satellite Communications in the New Space Era: A Survey and Future Challenges. , 70–109 pages. https://doi.org/10.1109/COMST.2020.3028247

[37] Jean Michel Lasry and Pierre Louis Lions. 2007. Mean field games. *Japanese Journal of Mathematics* 2, 1 (2 2007), 229–260. https://doi.org/10.1007/s11537-007-0657-8

[38] Mathieu Laurière and Ludovic Tangpi. 2022. Convergence of large population games to mean field games with interaction through the controls. *arXiv* (3 2022). http://arxiv.org/abs/2004.08351

[39] Wei Li, Luliang Jia, Quan Chen, and Yingwu Chen. 2024. A Game Theory-Based Distributed Downlink Spectrum Sharing Method in Large-Scale Hybrid Satellite Constellations. *IEEE Transactions on Communications* 72, 8 (2024), 4620–4632. https://doi.org/10.1109/TCOMM.2024.3375813

[40] Xin Lin, Aijun Liu, Chen Han, Xiaohu Liang, Kegang Pan, and Zhixiang Gao. 2023. LEO Satellite and UAVs Assisted Mobile Edge Computing for Tactical Ad-Hoc Network: A Game Theory Approach. *IEEE Internet of Things Journal* 10, 23 (12 2023), 20560–20573. https://doi.org/10.1109/JIOT.2023.3299950

[41] Pierre Louis Lions and Jean Michel Lasry. 2007. Large investor trading impacts on volatility. *Annales de l'Institut Henri Poincare (C) Analyse Non Lineaire* 24, 2 (2007), 311–323. https://doi.org/10.1016/j.anihpc.2005.12.006

[42] Mark W. Maier, Frank W. Gallagher, Karen St. Germain, Richard Anthes, Cinzia Zuffada, Robert Menzies, Jeffrey Piepmeier, David Di Pietro, Monica M. Coakley, and Elena Adams. 2021. Architecting the future of weather satellites. *Bulletin of the American Meteorological Society* 102, 3 (2021), E589–E610. https://doi.org/10.1175/BAMS-D-19-0258.1

[43] Outer Space Institute. 2026. *CRASH Clock*. https://outerspaceinstitute.ca/crashclock/ CRASH Clock value listed as 3.0 days on 20 March 2026.

[44] Pierre Degond, Michael Herty, and Jian-Guo Liu. 2017. Meanfield Games and Model Predictive Control. *Communications in Mathematical Sciences* 15, 5 (2 2017), 1403–1422. https://intlpress.com/JDetail/180626499443621889

[45] Tomas Pippia, Valentin Preda, Samir Bennani, and Tamas Keviczky. 2022. Reconfiguration of a satellite constellation in circular formation orbit with decentralized model predictive control. *arXiv* (1 2022). http://arxiv.org/abs/2201.10399

[46] Yongtao Su, Yaoqi Liu, Yiqing Zhou, Jinhong Yuan, Huan Cao, and Jinglin Shi. 2019. Broadband LEO satellite communications: Architectures and key technologies. *IEEE Wireless Communications* 26, 2 (4 2019), 55–61. https://doi.org/10.1109/MWC.2019.1800299

[47] Kathiravan Thangavel, Roberto Sabatini, Alessandro Gardi, Kavindu Ranasinghe, Samuel Hilton, Pablo Servidia, and Dario Spiller. 2023. Artificial Intelligence for Trusted Autonomous Satellite Operations. *Progress in Aerospace Sciences* 144 (12 2023). https://doi.org/10.1016/j.paerosci.2023.100960

[48] Kathiravan Thangavel, Dario Spiller, Roberto Sabatini, Stefania Amici, Nicolas Longepe, Pablo Servidia, Pier Marzocca, Haytham Fayek, and Luigi Ansalone. 2023. Trusted Autonomous Operations of Distributed Satellite Systems Using Optical Sensors †. *Sensors* 23, 6 (3 2023). https://doi.org/10.3390/s23063344

[49] Dezhi Wang, Wei Wang, Yuhan Kang, and Zhu Han. 2022. Dynamic Data Offloading for Massive Users in Ultra-dense LEO Satellite Networks based on Stackelberg Mean Field Game. In *IEEE INFOCOM WKSHPS: PerAI-6G 2022: Pervasive Network Intelligence for 6G Networks*. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798204

[50] Dezhi Wang, Wei Wang, Yuhan Kang, and Zhu Han. 2023. Distributed Data Offloading in Ultra-Dense LEO Satellite Networks: A Stackelberg Mean-Field Game Approach. *IEEE Journal of Selected Topics in Signal Processing* 17, 1 (2 2023), 112–127. https://doi.org/10.1109/JSTSP.2022.3226400

[51] Lixiang Wang, Dong Ye, Xianren Kong, Ming Liu, and Yan Xiao. 2024. Decentralized Receding Horizon Control for Satellite Cluster Reconfigurations With Successive Convexification Method. *IEEE Trans. Aerospace Electron. Systems* 60, 5 (2024), 5920–5936. https://doi.org/10.1109/TAES.2024.3398607

[52] Yao Wang, Chungang Yang, Tong Li, Xinru Mi, Lixin Li, and Zhu Han. 2024. A Survey on Mean-Field Game for Dynamic Management and Control in Space-Air-Ground Network. *IEEE Communications Surveys and Tutorials* 26, 4 (2024), 2798–2835. https://doi.org/10.1109/COMST.2024.3393369

[53] Yang Wu, Guyu Hu, Fenglin Jin, and Jiachen Zu. 2019. A Satellite Handover Strategy Based on the Potential Game in LEO Satellite Networks. *IEEE Access* 7 (2019), 133641–133652. https://doi.org/10.1109/ACCESS.2019.2941217

[54] Zhuochen Xie, Lu Ma, and Xuwen Liang. 2012. Unlicensed spectrum sharing game between LEO satellites and terrestrial cognitive radio networks. *Chinese Journal of Aeronautics* 25, 4 (8 2012), 605–614. https://doi.org/10.1016/S1000-9361(11)60425-1

[55] Jiapeng Xu, Xiang Chen, Ying Tan, and Guoxiang Gu. 2024. Robust Mean-Field Games With Partial Observations: A Complementary Strategy. *IEEE Trans. Automat. Control* 69, 12 (2024), 8766–8773. https://doi.org/10.1109/TAC.2024.3419002

[56] Yun Xu, Yulin Zhang, and Li Fan. 2024. Autonomous semi-major axis adjustment for mega constellation continuous coverage. *Advances in Space Research* 73, 11 (6 2024), 5582–5594. https://doi.org/10.1016/j.asr.2023.07.016

[57] Weiyi Yang, Yingwu Chen, Xiaolu Liu, Jun Wen, and Lei He. 2025. Distributed Satellites Dynamic Allocation for Grids with Time Windows: A Potential Game Approach. *Journal of Cleaner Production* (3 2025), 1–19. http://arxiv.org/abs/2503.08385

[58] Yafeng Zhan, Peng Wan, Chunxiao Jiang, Xiaohan Pan, Xi Chen, and Song Guo. 2020. Challenges and Solutions for the Satellite Tracking, Telemetry, and Command System. *IEEE Wireless Communications* 27, 6 (12 2020), 12–18. https://doi.org/10.1109/MWC.001.2000089

[59] Peixin Zhang, Wenting Wei, Kun Wang, Lizhe Liu, Liang Qin, and Celimuge Wu. 2024. Coalition game-based clustering algorithm for LEO satellite networks. In *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*. IEEE Global Communications Conference, 5042–5047. https://doi.org/10.1109/GLOBECOM52923.2024.10901516

[60] Tao Zhang, Yiji Zhu, Dongying Ma, Chaoyong Li, and Xiaodong Wang. 2024. Toward Rapid and Optimal Strategy for Swarm Conflict: A Computational Game Approach. *IEEE Trans. Aerospace Electron. Systems* 60, 3 (6 2024), 3108–3120. https://doi.org/10.1109/TAES.2024.3361436

[61] Liancheng Zheng, Xuemei Wang, Feng Li, Zebing Mao, Zhen Tian, Yanhong Peng, Fujiang Yuan, and Chunhong Yuan. 2025. A Mean-Field-Game-Integrated MPC-QP Framework for Collision-Free Multi-Vehicle Control. *Drones* 9, 5 (5 2025). https://doi.org/10.3390/drones9050375

[62] Renjun Zheng, Haibo Wang, Matthieu De Mari, Miao Cui, Xiaoli Chu, and Tony Q.S. Quek. 2021. Dynamic Computation Offloading in Ultra-Dense Networks Based on Mean Field Games. *IEEE Transactions on Wireless Communications* 20, 10 (10 2021), 6551–6565. https://doi.org/10.1109/TWC.2021.3075028

[63] Zixuan Zheng, Jian Guo, and Eberhard Gill. 2018. Onboard mission allocation for multi-satellite system in limited communication environment. *Aerospace Science*

and Technology 79 (8 2018), 174–186. https://doi.org/10.1016/j.ast.2018.05.022

[64] Zejian Zhou and Hao Xu. 2022. A Novel Mean-Field-Game-Type Optimal Control for Very Large-Scale Multiagent Systems. *IEEE Transactions on Cybernetics* 52, 6 (6 2022), 5197–5208. https://doi.org/10.1109/TCYB.2020.3028267

Satellite scheduling optimization for BepiColombo STC channel target-phase

Alberto Avallone
University of Modena and Reggio Emilia - Department of Sciences and Methods for Engineering (DISMI)
Reggio Emilia, Italy
alberto.avallone@unimore.it

Francesco Sala
University of Modena and Reggio Emilia - Department of Sciences and Methods for Engineering (DISMI)
Reggio Emilia, Italy
253589@studenti.unimore.it

Maxence Delorme
Tilburg University - Tilburg School of Economics and Management (TiSEM)
Tilburg, The Netherlands
m.delorme@tilburguniversity.edu

Cristina Re
Italian National Institute of Astrophisics (INAF) - Astronomic Observatory of Padua (OAPd)
Padua, Italy
cristina.re@inaf.it

Manuel Iori
University of Modena and Reggio Emilia - Department of Sciences and Methods for Engineering (DISMI)
Reggio Emilia, Italy
manuel.iori@unimore.it

ABSTRACT

The ESA-JAXA *BepiColombo* mission is a European-Japanese effort aimed at exploring Mercury, scheduled to reach the planet in 2027. Among its multiple instruments is the Stereo Imaging Channel (STC), a dual-camera that, after a global mapping of the planet, will acquire medium-resolution color images of different targets on the hermean surface. Planning STC target observations is challenging due to the limitations in data volume and operations. It requires a careful optimization of surface observations, considering the scientific priority of each target. The scheduling problem is formulated as a mixed-integer linear programming model and addressed using greedy heuristic algorithms. The results demonstrate that optimal, constraint-compliant solutions can be generated with the model, whereas the heuristics produce moderate-quality solutions within minutes of computation. The present work represents an initial phase to develop an autonomous agent for managing STC acquisition.

KEYWORDS

Scheduling, Optimization, BepiColombo

ACM Reference Format:
Alberto Avallone, Francesco Sala, Maxence Delorme, Cristina Re, and Manuel Iori. 2026. Satellite scheduling optimization for BepiColombo STC channel target-phase. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 4 pages.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

1 INTRODUCTION

The BepiColombo mission [2] has been developed as a joint effort between the European Space Agency (ESA) and the Japan Aerospace eXploration Agency (JAXA). Its nominal science phase is scheduled to begin in 2027 and its primary objective is to study the planetary surface of Mercury.

The European suite, called *Mercury planetary orbiter*, is equipped with multiple instruments such as the Spectrometer and Imagers for Mercury planetary orbiter BepiColombo Integrated Observatory SYStem (SIMBIO-SYS), an integrated imaging system composed of three complementary scientific channels: the Visible Infrared Hyperspectral Imager Channel (VIHI), the High Resolution Channel (HRIC), and the Stereo Imagining Channel (STC) [3]. The STC mission is divided into two phases: a global mapping of Mercury's surface over two aphelions (here, an *aphelion* is defined as a predetermined period during a hermean year when the planet is farthest from the Sun), followed by a target acquisition of specific areas over two additional aphelions. The STC is a camera system composed of two optical heads, called *channels*. Each channel has two color filters (for a total of four), used during the target acquisition phase, and one panchromatic filter used during the global mapping phase.

The present work focuses on the second phase of STC operations. We modeled the task of acquiring specific areas as an image scheduling problem, a well known problem in the literature [4], and addressed it using a range of optimization techniques.

In Section 2, the problem is described from an operational perspective. Section 3 presents a complete planning approach, including the preprocessing of input data. Sections 4 and 5 describe the proposed solution techniques, while preliminary results are presented in Section 6. Finally, conclusions are given in Section 7.

2 PROBLEM DESCRIPTION

The objective of the second phase of the STC mission is to cover all possible target areas, ranked according to their scientific relevance. The targets can be acquired by the STC during one of two distinct periods when Mercury is the furthest from the Sun, known as *aphelions*. The STC mission will take place during the third and

the fourth aphelions, each lasting approximately thirty earth-days (the first two aphelions being used for a global mapping phase). A target is considered covered when its entire area is acquired with all four color filters. Each STC channel image (also referred to as a *frame*) has fixed dimensions determined prior to acquisition. Because the field of view of a single frame is typically smaller than the target area, multiple frames must be acquired. To merge them into a single image, a certain degree of overlap between consecutive frames is required. In this study, it was agreed with the SIMBIO-SYS team to set the minimum overlap to 15%. A contiguous sequence of frames captured within the same orbit that meets this overlap requirement is defined as an *observation*. For larger targets, multiple observations may be required. In such cases, an inter-observation overlap constraint is enforced: for each pair of (selected) consecutive observations covering the target, the intersection must cover at least 15% of the area of the smaller observation. A *pattern* is defined as a set of observations that collectively cover a target area while satisfying all internal (along-track) and external (cross-track) overlap constraints.

There are two main constraints on the BepiColombo spacecraft. The first is the data volume generated by the acquisitions, which is limited both per earth-day (given the average estimated downlink data rate) and over the entire mission, as agreed with the SIMBIO-SYS team. In this work, we assume that this daily data limit remains constant. The second constraint limits the total number of spacecraft operations, each of which is referred to as a telecommand (TC). Hosting numerous instruments, BepiColombo must balance energy consumption by limiting the number of TCs per orbit for each instrument. Each STC observation requires a fixed set of TCs, which is the same regardless of the number of frames in the observation. Our goal is to maximize the scientific value of all acquired targets while respecting the spacecraft's physical and operational constraints.

Furthermore, the SIMBIO-SYS team has defined a strategic operation, called *fusion*, to save TCs in a given orbit. Indeed, two observations can be merged into one by keeping the STC channels continuously switched on. This approach halves the number of required TCs, since each observation has the same TC cost regardless of its length, but increases the total data volume, as some frames acquired during a fusion may cover areas that are not scientifically valuable.

3 PLANNING PIPELINE

This work presents the initial phase in the development of an autonomous agent for managing the STC mission acquisitions, which consists of evaluating various strategies to identify the most effective approach for integration into the agent.

The framework takes as input the shapefiles generated by the Simulator for Operation of Imaging Missions (SOIM) [7], a tool developed within the SIMBIO-SYS team to simulate the expected instrument pointing at specific time instances for each filter, with a minimum time interval of one second. Each frame is represented as a shapefile, generated with a time step chosen to ensure at least 15% overlap along-track. The output consists of the set of TCs to be sent to the spacecraft to initiate its operations, along with the shapefiles of the resulting coverage. These can be visualized in a geographic information system visualization tool using the SOIM outputs.

In our pipeline, we utilize the convex-hull area of the target, as shown in Figure 1. This approach allows for a fast identification of the intersection area between frames and targets, ensuring that only one (the most complete) sequence of frames is generated per target per orbit, thereby avoiding an increase in the number of TCs for irregular targets.

The framework follows a four-step pipeline: preprocessing, observation generation, single-channel integration, and solution techniques. In the following, we describe the first, second, and third steps, while the fourth step, which involves the development of dedicated optimization algorithms, is detailed in the next section.

3.1 Preprocessing

During the preprocessing phase, all frames that cannot be acquired due to the Sun's influence are excluded. If the Sun elevation angle over the horizon is outside the range $0°–180°$ (i.e., outside a hermean day), spacecraft operations cannot be performed due to the lack of light. To select admissible observations, we matched the possible frames with the convex hull of each target. The set of admissible frames is then divided into two groups: frames that cover at least part of a target and frames that do not cover any target. The latter group is used only when fusing observations, if necessary.

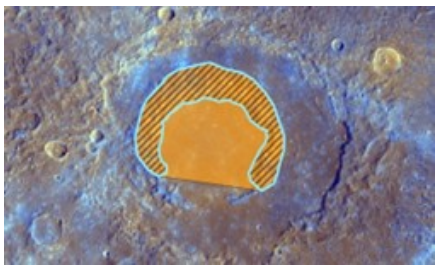


Figure 1: The original target (striped area) and the added area (not striped) which creates the convex hull

3.2 Observation generation

The observation generation phase focuses on identifying observations by grouping consecutive frames from previously selected sequences of filtered frames, as shown in Figure 2. Note that an observation may include consecutive frames covering two different targets simultaneously (see, e.g., the third observation from the right in the figure). Considering the variation in the equivalence between pixels and meters while moving towards the poles, we developed a dynamic filtering mechanism to remove unneeded frames for each observation. This mechanism determines the best possible time step while ensuring that the overlap constraint is respected.

3.3 Single-channel integration

Each STC channel acquires observations with both filters simultaneously, therefore, observations are merged in pairs of filters belonging to the same channel.

Table 1: Sets definitions

Name	Description
G	Set of all targets that can be acquired by STC
C	Set containing the two channels of STC
O	All the orbits performed by BepiColombo
D	Day of the year as recorded on earth
P_{gc}	Set of patterns covering the target $g \in G$ for the channel $c \in C$
Q_p	All the observations which belongs to a specific pattern $p \in P_{gc}$
Q_{oc}	Observations acquired by channel $c \in C$ in orbit $o \in O$
E_{oc}	Set of all fusion segments $e = (q_1, q_2)$ where $q_1, q_2 \in Q_{oc}$ are subsequent observations

Table 2: Parameters and variables definitions

Name	Description
s_g	Scientific value of the target $g \in G$
B	Maximum number of TCs for every orbit $o \in O$
V_d	Maximum data volume for each day $d \in D$
V_{tot}	Maximum data volume over the entire mission
x_q	Decision variable taking the value 1 if the observation $q \in Q_p$ is acquired, 0 otherwise
y_p	Decision variable taking the value 1 if the pattern $p \in P_{gc}$ is acquired, 0 otherwise
z_g	Decision variable taking the value 1 if the target $g \in G$ is acquired, 0 otherwise
x'_e	Decision variable taking the value 1 if the fusion observation $e \in E$ is acquired, 0 otherwise

4 SOLUTION TECHNIQUES

After the preprocessing and observation generation phases, the resolution step is applied. The task can be modeled as a weighted maximal coverage problem, which is a variation of the classical set coverage problem [6]. In our previous studies on satellite scheduling problems [1, 5], a mathematical model was defined and several heuristics were introduced. Building on these earlier works, we introduce here a mixed-integer linear programming (MILP) model and a greedy heuristic algorithm. A summary of the mathematical notation used in this section is presented in Tables 1 and 2.

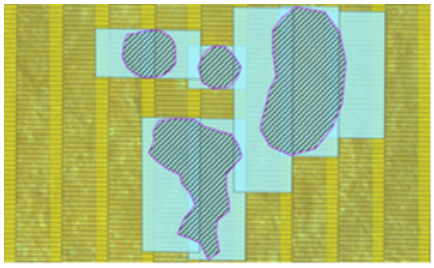


Figure 2: Generated feasible observations (cyan) over the raw sequences of frames (yellow) covering the targets (areas within the purple lines)

4.1 Mathematical model

The problem is defined as an MILP model that aims to maximize the accumulated scientific rank s_g of the acquired targets over all $g \in G$ possible targets. A target g is considered acquired ($z_g = 1$) only if the model selects a pattern $p \in P_{gc}$ that covers g with each channel $c \in C$. A pattern p can be selected ($y_p = 1$) only if all its observations $q \in Q_p$ are selected ($x_q = 1$). As defined in Section 2, the model must satisfy specific operational constraints. For each orbit $o \in O$, the total number of selected observations must not exceed the maximum number of TCs, B. Furthermore, for each day $d \in D$ and over the entire mission, the data volume of the selected observations must not exceed, respectively, the maximum daily capacity V_d and the cumulative data limit V_{tot}. To speed up the model resolution, we introduce a filtering mechanism that removes *dominated patterns*.

4.1.1 Dominant pattern selection . The dominant pattern selection method eliminates all patterns that are *dominated* by another valid pattern. A pattern $p' \in P_{gc}$ is dominated by a pattern $p'' \in P_{gc}$ if p'' is a subset of p' for the same target. Such dominated patterns should never be selected, as they increase data consumption without providing additional coverage compared to their dominating counterparts. By identifying and removing these patterns, the number of generated patterns for each target is reduced, which in turn lowers the computational complexity. This selection process is illustrated in Figure 3.

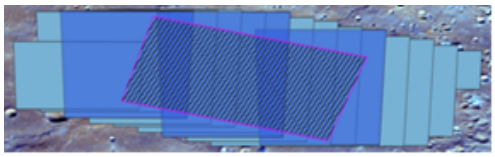


Figure 3: The selected coverage with 3 observations (dark blue) compared to 15 possible (light blue) covering the target (area within the purple line)

4.2 Greedy heuristic observation selection

The general framework of greedy heuristic algorithms can be described as follows. First, the available data volume and number of TCs are initialized, and the targets are sorted according to their scientific value. Then, starting with the highest-ranking target, the framework attempts to generate a feasible coverage pattern that minimizes cross-track overlap among observations. To construct the pattern, observations are first sorted and then added iteratively, provided no constraints are violated, until the target is fully covered. We implemented two sorting methods for the observations:

- *Orbit order search:* Observation are sorted by ephemeris time of execution. The process begins by selecting the last observation that covers the first edge of the target, and then iteratively selects subsequent observations that minimize overlap while adhering to the 15% threshold.
- *Point area search:* Observations are sorted by the proportion of the target area they cover. The process begins by selecting the observation that covers the largest portion of the target (typically located at the center of the target), and then iteratively selects subsequent observations that minimize overlap while adhering to the 15% threshold, proceeding first to the left and then to the right of the initial observation.

Note that, since the solution is constructed iteratively, it cannot be guaranteed that a pattern will be fully acquired, as the data and TC constraints may result in only partial coverage of some targets. The scientific value s_g of a target is collected only if the target is fully acquired.

5 FUSION

A particularly important consideration for the SIMBIO-SYS team is the trade-off between memory consumption and TCs, with memory being relatively abundant, while TCs are expected to be limited, as the satellite serves multiple scientific missions, each of which also requires TCs.

To allow fusion in the model, we enumerate every pair of consecutive observations (which, by construction, cover distinct targets) having the same orbit $o \in O$ and channel $c \in C$ and add the corresponding fusion segment into the set E_{oc}. If a fusion segment $e \in E_{oc}$ is selected, the TC count is reduced by two, and both the daily and total data volume are increased according to the size of the segment.

The fusion concept has not yet been tested for the MILP model, but we integrated it into the greedy heuristic as follows: when checking whether an observation satisfies all constraints (before adding it to a pattern), if it violates the TC constraint but not the memory constraint, we attempt to find the two observations in the same orbit whose fusion would minimize the increase in data volume, and fuse them.

6 RESULTS

The model-based ("Exact") and greedy heuristic ("Greedy") solution methods were tested to evaluate their operational performance, as summarized in Table 3, using the list of targets defined for the STC mission during the third aphelion. Both methods were implemented in Python and given a time limit (TL) of 3600 s. The model was solved to optimality using the commercial solver GUROBI. All computations were performed on a computer equipped with an AMD Ryzen 3700X CPU and 32GB of RAM, running Windows 11. The model was tested both with the dominant pattern selection approach ("Dominant") and without it ("No Filter"). The heuristic was tested in three configurations: orbit order search ("Orbit"), point area search ("Point"), and a combination of orbit order search and observation fusion ("Orbit-Fusion"). For each model, we report the number of variables ("#Var") to assess the effectiveness of the preprocessing, as well as the upper bound ("UB") value at which the solver terminates. For context, the third aphelion includes 195 targets (out of 404 in total) with a cumulative scientific value of 701.

Table 3: Results for Aphelion 3 instance

Method	Type	#Var.	Objective	UB	Time(s)
Exact	No Filter	1 093 368	678	699	TL
Exact	Dominant	7156	678	678	2162*
Greedy	Orbit	-	533	-	15
Greedy	Point	-	530	-	17
Greedy	Orbit-Fusion	-	543	-	82

*Excluding filtering time (12.83 s)

We observe that, without pattern filtering, the model cannot be solved to optimality within the time limit because of the very large number of generated patterns (and, consequently, variables). Applying the filtering mechanism has a significant impact, reducing the number of variables by 99.35% and allowing the model to be solved within the time limit. Regarding the heuristics, their performance is mixed. Although they are relatively fast, they produce solutions of moderate quality, with an average optimality gap of around 20%. Among the three heuristics tested, "Orbit-Fusion" achieves slightly better results but also requires the longest computation time.

7 CONCLUSION

In the present work, the BepiColombo STC mission problem is defined, and both a mathematical model and a set of greedy heuristics have been developed. Whereas the greedy heuristic reached only moderate-quality solutions, their fast running times and the fact that they do not require a commercial solver make them especially useful tools for gauging the viability of planning strategies for the identified targets. They also allowed for an estimation of the potential gains offered by the fusion technique; although computationally more demanding, this approach appears to be highly beneficial. We are currently working to integrate this process into the SIMBIO-SYS planning pipeline for application across the entire mission as an autonomous planning tool. We also plan to develop a metaheuristic that is expected to be both faster than the exact model and more effective than the current heuristics, as well as to test an extension of the model incorporating the fusion technique. We will further apply our methods to an extended set of 650 targets recently provided by the SIMBIO-SYS team. Finally, we plan to analyze potential future deployment conditions to perform a sensitivity analysis and evaluate the operational capability of our algorithms.

ACKNOWLEDGMENTS

We gratefully acknowledge financial support by NEXT Ingegneria dei Sistemi S.p.A and by the Italian Ministry of University and Research, under project PRIN PNRR 2022, grant n. P2022XF72W, "Combining mAchine Learning and optImization for Planetary remote Sensing missiOns (CALIPSO)".

REFERENCES

[1] A. Avallone, B. Ferrari, A. Cicchetti, M. Delorme, M. Iori, M. Lippi, and R. Orosei. 2025. Stochastic Optimization of the MARSIS Observation Scheduling Problem using a Predict-then-Optimize Approach. In *Proceedings of the 14th International Workshop on Planning & Scheduling for Space (IWPSS 2025)*. Toulouse, France. Paper presented at IWPSS 2025; detailed program available online.

[2] J. Benkhoff and al. 2021. BepiColombo - Mission Overview and Science Goals. *Space Science Reviews* 217, 8 (2021), 90. https://doi.org/10.1007/s11214-021-00861-4

[3] G. Cremonese and al. 2020. SIMBIO-SYS: Scientific Cameras and Spectrometer for the BepiColombo Mission. *Space Science Reviews* 216, 5 (2020), 75. https://doi.org/10.1007/s11214-020-00704-8

[4] B. Ferrari, J.-F. Cordeau, M. Delorme, M. Iori, and R. Orosei. 2025. Satellite Scheduling Problems: A survey of applications in Earth and outer space observation. *Computers & Operations Research* 173 (2025), 106875. https://doi.org/10.1016/j.cor.2024.106875

[5] B. Ferrari, M. Delorme, M. Iori, M. Lippi, and R. Orosei. 2025. *A Predict-then-Optimize Approach for the Research of Underground Water on Mars.* Technical Report. University of Modena and Reggio Emilia (UNIMORE).

[6] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming* 14, 1 (1978), 265–294. https://doi.org/10.1007/BF01588971

[7] E. Simioni, M. Zusi, and R. Politi. 2022. *SOIM User Manual v6.0.* Technical Report 202. INAF. https://doi.org/10.20371/INAF/TechRep/202

Multi-Satellite Observation Tasks Dispatching and Scheduling

Romain Barrault
DTIS, ONERA, Université de Toulouse
Toulouse, France
romain.barrault@onera.fr

Cédric Pralet
DTIS, ONERA, Université de Toulouse
Toulouse, France
cedric.pralet@onera.fr

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

Eric Sawyer
CNES
Toulouse, France
eric.sawyer@cnes.fr

ABSTRACT

A standard problem in the field of Earth observation is the scheduling of the observations of an agile satellite constellation. Given a set of end-user requests over Points Of Interest (POIs), it consists in selecting observations among the candidate ones, attributing each of them to a satellite, and defining the sequence of observations planned for each satellite given operational constraints. The latter are related to the visibility windows available to observe the POIs and the time-dependent maneuvers required to reorient the observation instrument between two POIs. They result in a highly combinatorial problem that must be solved in a restricted amount of time. To solve such a complex problem, we propose an approach that combines matheuristics (mathematical programming-based heuristics) to filter the observation tasks and metaheuristics to schedule them. Firstly, we solve a Sequential Ordering Problem for each satellite to get a giant tour visiting all the visible POIs. From this giant tour, we exploit a Linear Programming Model to compute the best observation set under several tour length constraints. Finally, we schedule the selected observations based on a Large Neighborhood Search. This three-step method substantially improves the solution quality when compared to a baseline scheduling approach.

KEYWORDS

Earth Observation Satellites, Constellation, Planning, Matheuristics, MILP, Large Neighborhood Search

ACM Reference Format:
Romain Barrault, Cédric Pralet, Gauthier Picard, and Eric Sawyer. 2026. Multi-Satellite Observation Tasks Dispatching and Scheduling. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 9 pages.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

1 INTRODUCTION

Earth observation constellations have become essential assets for a wide range of applications, from climate monitoring and disaster management to urban planning and security. The increasing number of satellites and the growing demand for timely, high-resolution data lead to a challenging planning and scheduling problem: each satellite must be assigned a sequence of observation tasks that respects strict temporal windows, resource constraints, and orbital dynamics while maximizing the overall scientific and economic value of the acquired imagery. This Multi-Satellite Earth Observation Scheduling Problem (MSEOSP) is intrinsically hard, as it can be formulated as a time-dependent Team Orienteering Problem with Time Windows (TD-TOP-TW) [10], where both the transition times and the profit of each observation vary with the exact acquisition time. Consequently, the problem is highly combinatorial (NP-hard), and highly sensitive to the interplay between multiple satellites, making exact solution methods impractical for realistic instance sizes and motivating the development of dedicated heuristic and metaheuristic methods.

In this paper, we propose a novel approach combining a matheuristic approach to select a subset of the (large) set of candidate observations for a constellation, and a metaheuristic scheduler, based on Large Neighborhood Search (LNS) [17, 18], to schedule the selected observations. The idea of the selector is to exploit the locations of the POIs and a simplified transition model in a Sequential Ordering Problem (SOP) solver in order to generate a so-called *giant tour* that then constrains a MILP aiming to select a set of observations maximizing the total reward. On this point, we note that giant tours are already used for vehicle routing problems [6], team orienteering problems [5], and even team orienteering problems with time windows [1]. In these related works, a unique giant tour visiting a subset of customers is split to get one tour per vehicle. Given that each satellite can see different POIs, the giant tour approach we propose is different: we build several giant tours (one per vehicle) taking into account some precedence constraints, and then select sub-tours of these giant tours to try and satisfy the temporal constraints. To further restrict which observations are selected based on the SOP ordering, we also consider sub-tours for sub-time frames to generate capacity constraints. Once a selection is provided by the selector, we use a LNS to schedule the observations, depending on their time windows and transition times.

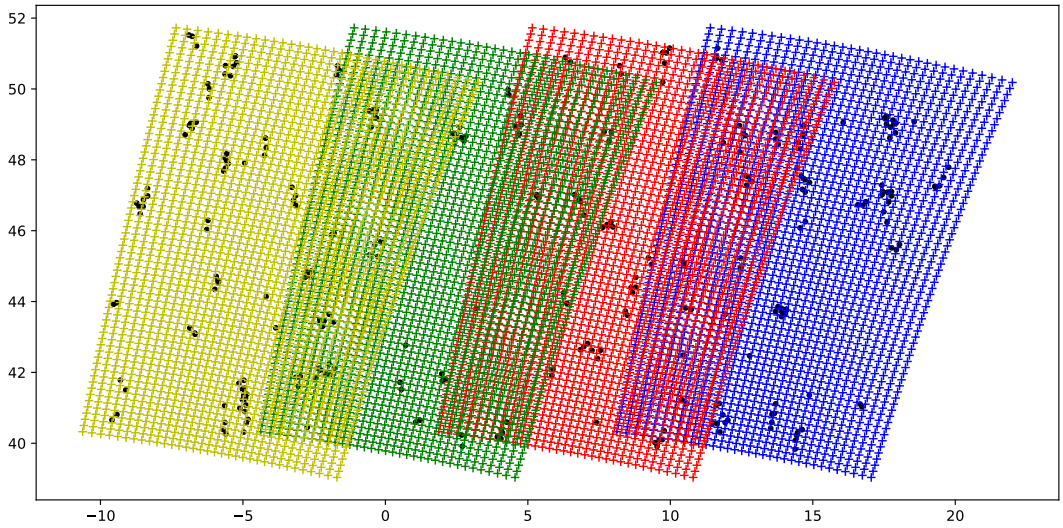


Figure 1: Sample problem instance represented in a Geographic Coordinates System: x-axis is latitude, y-axis is longitude. Black dots correspond to POIs. The colored meshings illustrate the areas successively overflight by four different satellites evenly separated on the same orbit, with a shift of the area overflight between two successive satellites due to the rotation of Earth.

The remainder of the paper is structured as follows. Section 2 retraces previous and current works on similar problems in the literature. Section 3 presents the core concepts and formalizes the problem. Section 4 overviews the global approach we follow. Next sections zoom in the main building blocks of our method: a matheuristic selector combining sequential ordering problem solving and an integer linear program that selects candidate observations to schedule, in Section 5, and a metaheuristics scheduler based on a large neighborhood search aimed to schedule the selected observations, in Section 6. Our approach is experimentally evaluated on a constellation composed of four agile low-orbit satellites, on scenarios where the meteorological conditions (cloud coverage) impact the quality of the observations. We compare our approach to a baseline LNS running for 20 minutes in Section 7. Finally, Section 8 concludes the paper with perspectives.

2 BACKGROUND

The Multi-Satellite Earth Observation Scheduling Problem is a Time-Dependent Team Orienteering Problem with Time Windows and Time-Dependent Profit (TD-TOP-TW-TDP). The latter is an extension of TD-TOP-TW, where the goal is to design a set of routes for a fleet of vehicles to maximize the total collected profit while respecting a total time budget, time windows at visited locations, and time-dependent travel times, while maximizing the sum of time-dependent profits (*e.g.*, profit increasing with service time) [14]. Due to its computational complexity, exact methods, such as Mixed Integer Linear Programming (MILP) formulations, are typically only viable for small-scale instances [14]. For larger, more realistic scenarios, metaheuristics are the dominant solution approach. Effective metaheuristic techniques include Variable Neighborhood Search (VNS) [14], Iterated Local Search (ILS) [11], and nature-inspired algorithms like the Hybrid Artificial Bee Colony (HABC) algorithm, which are designed to balance exploration and exploitation of the search space [20]. These methods often employ tailored local search operations and insertion heuristics to efficiently handle the dynamic nature of travel times and the profit gained, which can depend on the vehicle's arrival and service time at a location [11]. Exact approaches based on Dynamic Programming (DP) and extensions of algorithms like A^* search have also been adapted for related time-dependent shortest path and TSP problems with time windows, often using concepts like partially time-expanded networks for improved performance [9].

In the space domain, Earth Observation Satellite Scheduling Problems require the maximization of total priority/profit from observation requests while respecting time windows (target visibility periods) and intricate operational constraints (e.g., memory capacity, power availability, slew angle/maneuver time, and non-overlap of tasks) [7]. The core techniques mirror those for the TD-TOP-TW, centered around Mixed-Integer Linear Programming (MILP) for optimal solutions on small instances, and metaheuristics for large-scale real-world problems. Reinforcement learning techniques have also recently proven to be competitive in the context of multi-satellite scheduling [16, 19]. For large constellations or long planning horizons, decomposition methods and hybrid metaheuristics are crucial, often combining Local Search (like Iterated Local Search or Simulated Annealing) with greedy construction heuristics and calls to Constraint Programming (CP) solvers for solving sub-problems[2, 12]. MILP formulations often represent the problem using a time-space network or as a variation of the Traveling Salesman Problem with Time Windows (TSPTW) or Orienteering Problem [15]. The next section details the problem's features.

3 PROBLEM DEFINITION

We consider the problem of scheduling tasks for a low orbit agile EOS constellation. In this problem, observation requests over

specific POIs are submitted to the system. Each satellite of the constellation overflies different candidate POIs along its orbit, but corresponding areas may overlap, implying a need to assign each observation to only one satellite (see Fig. 1). The observations also come up with an individual reward which depends on a cloud cover forecast and the time at which each satellite can observe the corresponding POI (time-dependent profit). The purpose is to compute the best constellation schedule, that means the set of individual schedules for each satellite which maximizes the total sum of rewards collected by the constellation.

3.1 Input data: satellite observation areas

We define our case study in order to measure the efficiency of the allocation part of our method without drastically increasing the problem complexity. Firstly, we define input data related to a single agile satellite that can point left, right, forward, or backward thanks to gyroscopic actuators. For this satellite, we consider a time horizon $[0, H_0]$ where $H_0 = N \cdot \delta_T$ is defined from a timestep δ_T and a number steps N. Then, we determine the Earth portion visible by the satellite during time frame $[0, H_0]$. This portion is centered on the satellite ground track and bounded in two dimensions: along track because of the restricted time horizon, and across track because of a maximum angle α_{max} beyond which observation quality is considered as too low. Any point within this area may be a candidate POI for the satellite (also called a target). For each target, we can also compute a Time Window (TW) during which the satellite can point to the target while having a visibility angle less than α_{max} (the farther the POI from the satellite ground track, the smaller the time window). Finally, each observation is associated with an individual reward in interval $[0.2, 1]$. This reward is obtained from a decreasing function of the local cloud cover forecast provided by real weather data.

Once the data for a single satellite is computed, we derive data for an entire EOS constellation with the following process. We define a set of satellites $\mathcal{S}$ along the same orbit, separated by a duration Δ_T. Basically, Δ_T is the time required for a satellite to reach the same orbital position as its predecessor. In our case, we consider a unique orbital plane and four satellites evenly spaced on this plane ($\Delta_T = 25$ minutes in the experiments). Satellite number s is considered over time frame $[s \times \Delta_T, s \times \Delta_T + H_0]$. From the set of Earth portions visible for the satellites, we get a set of potential candidate POIs.

3.2 Input data: candidate POIs within the observation areas

For each satellite s, we generate a set of N_s candidate POIs within its area, using some meshing to avoid having very close candidate POIs. The ith candidate POI for satellite s is referred to as $P_{s,i}$, and the visibility window of this POI by satellite s is denoted by $W_{s,i} = [Ostart_{s,i}, Oend_{s,i}]$. For this POI, we also compute an individual reward $R_{s,i}$. Note that in our settings, each candidate POI p may be viewed by several satellites, that is we may have $p = P_{s,i}$ for one satellite s and $p = P_{s',i'}$ for another satellite s'. However, due to the time-dependent cloud cover, the observation reward for p may depend on the satellite (*i.e.*, $R_{s,i}$ not necessarily equal to $R_{s',i'}$). In the following, we denote Π the set of positions for the POI visible by at least one satellite ($\Pi = \cup_{s \in \mathcal{S}, i \in [1..N_s]} P_{s,i}$). At this point, the set of candidate POIs makes up for an instance of our problem, as represented in Figure 1.

3.3 Optimization problem

As an input, we consider a set of POIs Π defining the so-called *Order Book*, and for each satellite s the set of candidate POIs ($\{P_{s,i} \mid i \in [1..N_s]\}$), their rewards ($\{R_{s,i} \mid i \in [1..N_s]\}$), and their time windows ($\{W_{s,i} \mid i \in [1..N_s]\}$). An additional input data comes from the time-dependent maneuvers between successive observations, in order for each satellite to point its observation instrument towards the right directions. We denote by $tt_{s,i,j,t}$ the duration of the maneuver by satellite $s \in \mathcal{S}$ when maneuvering from observation number $i \in [1..N_s]$ to observation number j, for a maneuver starting at time t. This transition function is time-dependent since it depends on the current position of the considered satellite on its orbit.

Definition 1. The *Multi-Satellite Earth Observation Scheduling Problem* (MSEOSP), for $\zeta = |\mathcal{S}|$ satellites, consists in finding ζ sequences of observations $\sigma_s = [\sigma_{s,1}, \ldots, \sigma_{s,K_s}]$ such that

- each candidate observation in Π appears at most once over all sequences in σ_s;
- for each satellite $s \in S$, all observations are performed during the available time window ($W_{s,i}$), while respecting the required transition times; formally, if number i is the first element of σ_s, the earliest start time for the corresponding observation is $t_{s,i} = Ostart_{s,i}$; if i and j are two consecutive elements in σ_s, then we have $t_{s,j} = \max(Ostart_{s,j}, t_{s,i} + tt(s, i, j, t_{s,i})$; finally, the transition constraints are satisfied if and only if condition $t_{s,i} \leq Oend_{s,i}$ is satisfied for every observation number i belonging to σ_s;
- the total collected reward $\sum_{s \in S, i \in \sigma_s} R_{s,i}$ is maximized.

The problem is usually over-constrained, that is it is usually not possible to observe all the candidate POIs.

4 GLOBAL SEARCH APPROACH

To solve the optimization problem presented before, we exploit a two-step approach involving a matheuristic that computes an estimation of the optimal solution of our problem given coarse-grain constraints, and a metaheuristic that adapts this solution given the detailed constraints. The global search architecture is illustrated in Figure 2.

4.1 Step 1: resolution of a Sequential Ordering Problem

First, for each satellite s, we compute a relevant observation order containing all the POIs it is able to see, in order to build a giant tour for s. Several methods can be used to get such a giant tour. A baseline method is the Earliest Start First (ESF, also referred to as *Earliest Due Date* [8]) heuristic that returns observations sorted by increasing window start times ($Ostart_{s,i}$), so that the satellite tries to observe any POI as soon as it is visible. In practice, this may lead to very bad solutions where along its orbit, the satellite is moving very often between the left and right parts of its ground track. This is why we exploit another method based on the resolution of a problem known as the *Sequential Ordering Problem* (SOP).

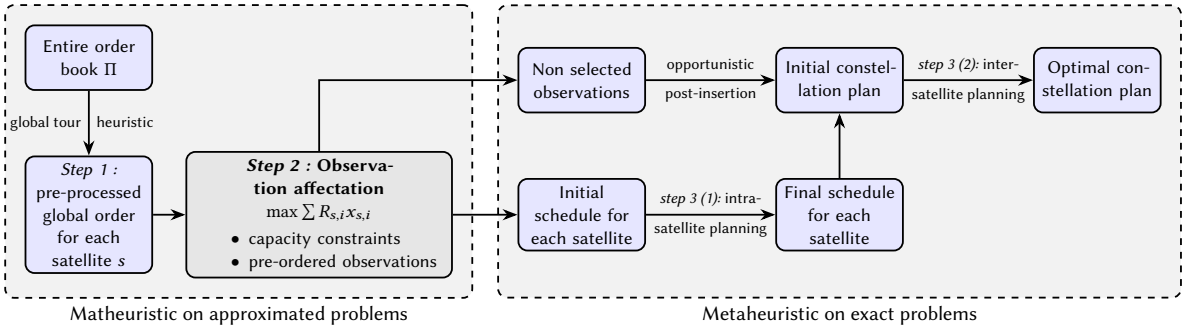


Figure 2: Constellation scheduling algorithm workflow

Definition 2. Given a weighted directed graph $G = (V, E)$ where we denote by $(c_e)_{e \in E}$ the set of costs over the edges of the graph, and a set of *precedence constraints* $\mathcal{P} \subset E$, the *Sequential Ordering Problem (SOP)* consists in the following optimization problem :

$$\min_{\sigma \subset E} \sum_{e \in \sigma} c_e \tag{1}$$

$$\text{s.t.} \quad \sigma = [(\sigma_1, \sigma_2), (\sigma_2, \sigma_3) ... (\sigma_{|V|-1}, \sigma_{|V|})] \tag{2}$$

$$\forall i \in [1, ..|V|], \sigma_i \in V \tag{3}$$

$$\forall v \in V, \exists! \, i \in [1, ..|V|] \; s.t. \; \sigma_i = v \tag{4}$$

$$\forall (v, v') \in \mathcal{P}, indexOf(v, \sigma) < indexOf(v', \sigma) \tag{5}$$

This can be viewed as an *Asymmetric Traveling Salesman Problem* with added precedence constraints implementing the impossibility for certain nodes to follow other ones in the resulting sequence (constraint (5)). There is however no time window for visiting the nodes, nor time-dependency on the transition costs, hence this is an approximated version of our problem.

4.2 Step 2: matheuristic to dispatch POIs to satellites

The SOPs solved provide us with a giant tour for each satellite. We then exploit these giant tours in a MILP model that assigns observations to satellites while taking into account the following constraints.

- Each POI may be assigned to at most one satellite.
- Given giant tour G_s for satellite s, the total cost associated with a selection of observations O_s for s corresponds to the total transition time (or cost) of the sub-tour of G_s induced by O_s. At this search step, this constraint allows us to get free of the decisions concerning POI observation ordering.
- The total time (or total cost) consumed by an observation sequence must be lower than a pre-determined value which is the duration between the earliest start time and the latest end time of candidate POI observations (overload checking, as in constraint programming methods). We also add local constraints on the observation sequence cost (more details later on this point).

To compute the total cost of a sequence while using only linear constraints in a MILP, we approximate the time-dependent transition times as time-independent transition times based on a specific policy, such as an *optimistic policy* considering the minimum transition duration among a set of possible discrete start times for the corresponding maneuver, or a *pessimistic policy* considering the maximum transition duration.

4.3 Step 3: metaheuristic part

At the end of the previous step, we obtain a good subset of candidate POIs assigned to each satellite. At that time, we resort to existing scheduling algorithms described in [3] to (1) plan observations without changing the assigned satellite first, and (2) then consider exchanging acquisitions between the satellites. The purpose is to first exploit the MILP result as a warm start for observation assignment, and then improve for the best the total plan quality. Basically, the existing scheduler introduced by Barrault et al. generates an initial plan thanks to a greedy method called `GreedySch` that iteratively inserts observations in the current plan. In this scheduler, the next observation to be inserted is selected based on heuristic values that depend on both the individual observation rewards and the additional maneuver durations required to insert observations. To improve the resulting plan, we resort to a *Large Neighborhood Search* algorithm, namely `LNS to schedule`, which iteratively destroys and repairs the current plan based on these same heuristic values. In the first part of the process, this algorithm is computed at the same time locally on all satellites, and in the second part a single `LNS to schedule` is performed to destroy and repair the plans in a multi-satellite context. This is done during a pre-determined amount of time, and in the end, the plan that got the best reward sum is returned.

5 MATHEURISTICS FOR SOLVING A RELAXED ROUTING PROBLEM WITH CAPACITIES

To quickly produce a first good quality solution, we resort to a *matheuristic* [4], that is to a technique that exploits mathematical programming techniques (MILP techniques in our case) to get heuristic solutions. The MILP model proposed in this context attempts to quickly produce good quality solutions by going beyond greedy heuristics that fail having a global view of the problem. Before detailing this MILP model, we present how we compute a global tour over all visible POIs from each satellite's perspective, based on the resolution of sequential ordering problems. The global tours

obtained are then exploited in a MILP model that avoids considering scheduling decisions and focuses on pure observation selection and assignment and then we refer to the obtained sequences to select and attribute observations from the order book with respect to satellite capacity constraints.

5.1 Sequential Ordering Problem (SOP)

In our process, the SOP solved can be seen as a problem relaxation where the decisions are focused on the routing part, that is on the transition times between observations, while the time windows available to perform the observations are (almost) completely discarded. This SOP is solved thanks to the LKH solver presented in [13]. For a satellite s that sees POIs numbered from 1 to N_s, as we are only looking for a sequence of observations and not a tour over them, we add a fictitious node numbered 0 which makes up for both the departure and the ending of our sequence. This gives us a weighted directed graph $G = ([0..N_s], E)$ where the edges are weighted by an approximation of the transition times between the observations. A key part in the problem definition is the establishment of a set of mandatory precedence constraints $\mathcal{P}_s$ between the observations of satellite s. More precisely, for two distinct observation numbers $i, j \in [1..N_s]$, if the maneuver towards j starting from i at time $Ostart_{s,i}$ arrives after the end of the window of j (after $Oend_{s,j}$), then j must precede i in the resulting sequence, that is

$$(Ostart_{s,i} + tt(s, i, j, Ostart_{s,i}) > Oend_{s,j}) \rightarrow ((j, i) \in \mathcal{P}_s) \quad (6)$$

The previous rule may however lead to both $(j, i) \in \mathcal{P}_s$ and $(i, j) \in \mathcal{P}_s$ for certain observation couples with very narrow and overlapping time windows. In this case, to avoid precedence cycles, only one of the two precedence constraints is kept in $\mathcal{P}_s$. We then compute all edge weights with the optimistic transition times policy.

Executing LKH over the SOP built provides us with an observation sequence that contains all the observations, satisfies the precedence constraints, and minimizes the total tour cost. Figure **??** displays the kind of sequence returned, for a satellite whose ground track traverses the center of the area from the top to the bottom. We can see on the figure that in the solution found, the satellite limits large maneuvers going backward with regards to its ground track. This is quite natural since backward maneuvers, requiring the satellite to move counter to its orbital motion, are usually longer for such a system.

5.2 MILP Model

From now on, for satellite s, the index $i \in [1..N_s]$ of a candidate observation corresponds to position of this observation in the order returned by the SOP solver. Moreover, $P_{s,i}$ still denotes the i-th candidate observation point of satellite s and Π denotes the set of candidate POIs over all the satellites. To select a subset of the candidate observations and assign each of them to a satellite, we implement the following MILP that takes into account some global capacity constraints related to the time windows. This model is detailed thereafter.

$$\text{maximize} \sum_{s \in \mathcal{S}} \sum_{i \in [1..N_s]} R_{s,i} \cdot x_{s,i} \quad (7)$$

s.t.

$$\forall s \in \mathcal{S}, \forall i \in [1..N_s],\ x_{s,i} = \sum_{j>i} next_{s,i,j} \quad (8)$$

$$\forall s \in \mathcal{S}, \forall i \in [1..N_s],\ x_{s,i} = \sum_{j<i} next_{s,j,i} \quad (9)$$

$$\forall s \in \mathcal{S},\ \sum_{j>0} next_{s,0,j} = 1 \quad (10)$$

$$\forall s \in \mathcal{S},\ \sum_{j \leq N_s} next_{s,j,N_s+1} = 1 \quad (11)$$

$$\forall p \in \Pi,\ \sum_{s \in \mathcal{S},\, i \in [1..N_s] \text{ s.t. } P_{s,i}=p} x_{s,i} \leq 1 \quad (12)$$

$$\forall s \in \mathcal{S}, \forall w \in \mathcal{W}_s,\ \sum_{(i,j) \in U} next_{s,i,j} \cdot T_{s,i,j} \leq Dur_w \quad (13)$$

where $U = \{(i, j) \in [1..N_s]^2 \mid i < j,\ W_{s,i} \subseteq w \vee W_{s,j} \subseteq w\}$

$$\forall s \in \mathcal{S}, \forall i \in [1..N_s], x_{s,i} \in \{0, 1\} \quad (14)$$

$$\forall s \in \mathcal{S}, \forall i, j \in [0..N_s + 1] \text{ s.t. } i < j,\ next_{s,i,j} \in \{0, 1\} \quad (15)$$

For each satellite $s \in \mathcal{S}$ and observation point $p = P_{s,i} \in \Pi$, $x_{s,i}$ is a Boolean variable that takes value 1 if and only if p is selected in the observation sequence of satellite s. Variables $next_{s,i,j}$ are Boolean variables equal to 1 if and only if:

- points $P_{s,i}$ and $P_{s,j}$ are both selected for satellite s,
- for each $k \in]i; j[$, $P_{s,k}$ is not selected for satellite s, that is $P_{s,j}$ is the next point selected after $P_{s,i}$ in the observation sequence returned by the SOP solver.

The objective function defined in Equation (7) corresponds to the total reward collected. Constraints (8) and (9) are standard flow constraints ensuring that each selected observation has a unique predecessor and successor in the sequence of selected observations. The constraints use two additional fictitious nodes: one node numbered 0 for the start of the observation plan, and on node numbered $N_s + 1$ for its end. Constraints (10) and (11) are particular cases for the start and end of the sequence. Constraint (12) models that each POI cannot be observed by more than one satellite.

Constraints (13) are capacity constraints over a set of specific time windows. More precisely, in these constraints, $T_{s,i,j}$ is the same approximated transition time between $P_{s,i}$ and $P_{s,j}$ as explained in Section 4. We also denote by $\mathcal{W}_s$ a set of time windows $w = [Wstart_w, Wend_w]$, of duration Dur_w, over which we attempt to prevent overloads in terms of transition times for satellite s. Set $\mathcal{W}_s$ is built as follows.

- The first added window is the larger one $\tilde{w}$: $\tilde{w} = [Wstart_{\tilde{w}}, Wend_{\tilde{w}}] = [\min_i Ostart_{s,i}, \max_i Oend_{s,i}]$ so that the resulting constraint concerns all observations for satellite s.
- We generate other windows iteratively. For this, we initiate an integer $Div = 2$. We create $nCand$ candidate windows of length $l = \frac{Wend_{\tilde{w}} - Wend_{\tilde{w}}}{Div}$. We set $nCand = 5$ after some experiments. The windows generated are spread in $\tilde{w}$, which means that they start at one of the timestamps $Wstart_{\tilde{w}} + (k-1) \cdot step$ for $k \in [1..nCand]$, where $step = \frac{Wend_{\tilde{w}} - l - Wstart_{\tilde{w}}}{nCand}$. Among all these windows w, we keep only those verifying $\exists i \in [1..N_s],\ W_{s,i} \subseteq w$. If there are still windows satisfying

this property, we increment Div and compute this step again, otherwise we stop the window generation process.

The set of capacity constraints given in Equation (13) limits the sum of transition times performed on numerous time windows.

In the end, the MILP obtained solves a time-independent version of the MSEOSP introduced in Section 1, using a pre-determined order and without taking into account detailed time windows. Solving this MILP gives us both an assignment of observations to satellites (based on the $x_{s,i}$ variables) and for each satellite, a pre-determined order on the selected set of observations (based on the $next_{s,i,j}$ variables) by following the path from node 0 to node $N_s + 1$. Of course, following the order provided by the global tour can be sub-optimal, but the goal is to define a MILP that can be quickly solved to get a first heuristic solution.

6 METAHEURISTIC FOR THE DETAILED SCHEDULING

In this section, we describe the greedy search and the Large Neighborhood Search (LNS) metaheuristic used to compute "actual" MSEOSP schedules over the satellites, where "actual" means that no input data of the MSEOSP is approximated from this point. More specifically, the time windows and the exact time-dependent transition times are both taken into account to get actually feasible solutions. Both the greedy search and LNS exploit operators that update the satellite plans at the level of a single satellite (Section 6.1), or at the level of the entire constellation (Section 6.2).

6.1 Intra-satellite Moves

When the previously described MILP model returns all observations selected for a given satellite, there is a need to obtain an actual schedule satisfying all the MSEOSP constraints. This part describes two different ways to do so.

6.1.1 Selection-based Greedy Repair Heuristic. This heuristic is described by algorithm 1. It starts either from an empy satellite schedule or one that is simply not full of observations, and fills this schedule according to a greedy policy that considers only the observations selected for the satellite in the MILP solution. The best observation to insert at any time is the one with the minimum insertion penalty. For this, the algorithm tests the addition of each unplanned observation at each possible position in the current schedule and computes an associated insertion penalty. The penalty value obtained for the insertion of observation $i \in [1..N_s]$ between two observations j and k in schedule σ_s is computed as:

$$\frac{tt(s, j, i, t_{s,j}) + tt(s, i, k, t_{s,i}) - tt(s, j, k, t_{s,j})}{R_{s,i}}$$

where $t_{s,j}$ describes the current observation time for j in σ_s and $t_{s,i} = t_{s,j} + tt(s, j, i, t_{s,j})$ gives the observation time of i after insertion (without considering the time window of i). The penalty value therefore takes into account both the temporal cost for inserting each acquisition at any position and the reward provided by the insertion. The downside of this method is that it ignores the formerly computed global order on $\sigma_s \cup \overline{\sigma_s}$. At any time during the process, the feasibility of an observation plan is verified using a `Earliest Start heuristic`.

Algorithm 1 Greedy Repair Heuristic

Require: `current schedule` σ_s
`unplanned observations` $\overline{\sigma_s}$

```
full ← false
while !full do
    p̃ ← ∞
    full ← true
    for acq ∈ σ̄_s do
        for position ≤ len(σ_s) do
            if insertable(i, position, σ_s) then
                full ← false
                p ← penalty(acq, position, σ_s)
                if p ≤ p̃ then
                    p̃ ← p
                    updateBestAcq(acq, position)
    if !full then
        insertBestAcq(σ, bestAcq, bestPosition)
        removeAcq(bestAcq, σ̄)
```

6.1.2 Tour-based Greedy Repair Heuristic. The second greedy heuristic is very similar to the former one except that it only considers schedules respecting the global order induced by the values obtained for the $next_{s,i,j}$ variables at the level of the matheuristic (cf. Section 5). In this so-called *Tour-based Greedy Repair Heuristic*, a single insertion position is tested at each step for each unplanned observation. This approach considers a restricted set of schedules but is must faster.

Randomized Greedy Destroy Heuristic. Once the greedy search process produced a full solution σ_s for each satellite s, the LNS part of the algorithm applies *destroy* and *repair* operations to iteratively enhance the schedule quality for s. The destroy operator used in this purpose removes K observations from the current schedule, with K randomly chosen between 1 and $|\sigma_s|/3$. At each destroy step, one observation is randomly removed from the schedule, with a higher removal probability for observations having a higher presence penalty. Here, we use the same penalty criterion as before: each observation in the current schedule has a presence penalty based on its reward and the time gained in case of removal.

6.2 Inter-satellites Moves

When all satellite schedules are optimized, the last part of the algorithm destroys and repairs the current solution at the constellation level. Also, any observation selected by the MILP solver that did not make it in the best schedule for the corresponding satellite after intra-sallite moves is considered as unworthy and will remain unused for the rest of the algorithm. Two ideas are exploited to try and get better rewards or smaller transitions: first, it can be relevant to reassign an observation to another satellite able to perform it, and second there may be some place left to insert observations not selected initially in the solution derived from the MILP. To tackle these two points, we define an Inter-Satellite Greedy Repair Heuristic (ISGRH) and a Randomized Greedy Destroy Heuristic (RGDH). These two heuristics are very similar to their intra-satellite counterpart. The only difference is that they browse all satellite current schedules before inserting or removing any observation, which

Table 1: Algorithm versions results on 72 instances: optimality gaps (%) at the end of any step. The best value per column is highlighted in gray. If bv is the best known reward on instance I with all algorithms, and algorithm A finds reward v on instance I at the end of step s, then the optimality gap (%) of A on I at the end of s is $100 \cdot (bv - v)/bv$.

Algorithm	initial solution				post intra-moves				post inter-moves			
	max	median	mean	min	max	median	mean	min	max	median	mean	min
Baseline (20min)	16.4	5.29	6.11	1.41	n/a	n/a	n/a	n/a	10.5	0.792	1.57	0
Baseline (1min)	16.4	5.29	6.11	1.41	n/a	n/a	n/a	n/a	10.5	1.35	1.85	0
ESF-SelGH	15.6	9.38	9.81	4.14	15.0	6.71	7.12	3.25	4.37	1.44	1.52	0
ESF-TourGH	24.0	14.8	14.8	8.66	15.5	6.81	7.13	3.25	7.91	1.60	1.87	0
LKH-SelGH	15.1	8.16	8.52	2.78	11.0	5.12	5.17	0.831	5.46	0.699	1.11	0
LKH-TourGH	20.2	10.2	10.7	2.35	10.3	4.79	5.07	0.892	6.80	0.754	1.17	0

implies that the lowest or highest penalty is computed over a wider set of observations and a wider set of insertion positions. Empirically, the inter-satellite moves and the insertion of observations not selected by the MILP allow us to enhance the final schedule quality, especially when problem approximations lead to some erroneous decisions at the MILP level. The counterpart of this scheduling process is that unlike intra-satellite moves, these computations cannot be performed in parallel. ISGRH and RGDH are also used to define our baseline LNS method. In this context, they start computations from an empty constellation schedule, and all observations can be inserted or removed since no MILP computation is involved. We have therefore presented all components of our algorithm shown in Figure 2. The next part describes its performances.

7 EXPERIMENTS

7.1 Instances

The instance set considers 36 different POI distributions. For each of them, 300 observation points are chosen in the area covered by the four satellites, as in Figure 1. In addition, in order to test our algorithms on various realistic study cases, we designed three instance archetypes in terms of POI local density :

(1) *dense* ones, where there are a handful of POI hubs with a high local density,
(2) *sparse* ones, where most POIs are isolated from one another,
(3) *mixed* ones, as a mix of both of these types, where POIs are either isolated or part of a very narrow hub.

To compute observation rewards, 24 real cloud cover scenarios from year 2024 are retrieved [1]. For a cloud cover scenario, the reward $R_{s,i}$ associated with an observation is equal to one in case of no cloud cover over point $P_{s,i}$, 0.2 for a full cloud cover, and quadratically decreasing in between. More precisely, the resulting reward function is a piecewise constant version of the latter one with 5 possible values. In the end, we obtain 36 different POI distributions and 24 cloud cover scenarios (corresponding to 1^{st} and 15^{th} of each month of 2024). To limit the size of the instance set, we consider only 2 cloud cover scenarios per POI distribution (and each cloud cover scenario is applied to exactly 3 POI distribution scenarios). This results in a total of 72 problem instances. Finally, visibility time windows are defined according to a maximum observation angle α_{max} set to 30° in accordance with operational needs.

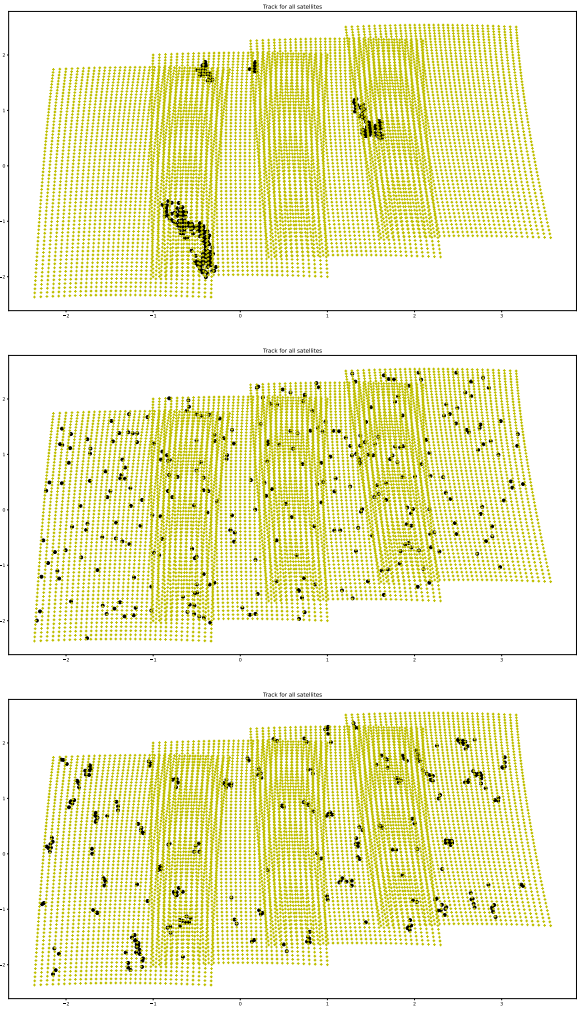


Figure 3: 2D representations of instance examples for each archetype described in 7.1. From top to bottom : *dense*, *sparse* and *mixed*. Black dots correspond to POIs and each yellow grid displays the Earth area covered by a single satellite.

[1] https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels

7.2 Experimental Setup

Firstly, two policies were implemented to return a global route over all satellite visible observations: the basic ESF heuristic that visits observation following their window start times, and the LKH heuristic that solves an SOP. Secondly, once the MILP solver returns a satellite-observation assignment, each detailed satellite schedule can be built based either on the Selection-based Greedy Repair Heuristic (denoted SelGH) or on the Tour-based Greedy Repair Heuristic (denoted TourGH). This leads to four different algorithm variants : ESF-SelGH, ESF-TourGH, LKH-SelGH, and LKH-TourGH.

The maximum CPU time for each algorithmic variant is set to 1 minute. A maximum CPU time of 40 seconds is dedicated to the matheuristic (including ESF or LKH, plus the resolution of the MILP model that is stopped when the optimality gap is less than 1%). The observation assignment problem is solved thanks to CPLEX solver, and the matheuristic usually converges much faster, typically in 5 seconds. Once the matheuristic produces a solution, at time t, the remaining time (1min minus t) is equally split between the intra-satellite LNS and the inter-satellite LNS. In the latter part, inter-satellite LNS operates with the possibility to insert observations that were not selected in the observation affectation process. Empirically, it allows us to enhance the final schedule's quality in case of unprecise choices in the latter part because of problem approximations. Thanks to problem decomposition, intra-satellite LNS is performed simultaneously over each satellite by multi-threading. All schedulers are implemented in Java and executed on 20-core Intel(R) Xeon(R) CPU ES-2660 v3 @ 2.60GHz, 62GB RAM, Ubuntu 18.04.5 LTS, with an OpenJDK 11.0.9 JVM.

The four algorithmic variants are compared to a baseline *LNS* algorithm (denoted Baseline) whose operators are ISGRH and RGDH.

7.3 Result Analysis

Performance metrics are shown in Table 1, where we display the rewards' optimality gap : (i) after initial greedy schedule generation for each satellite, that means with inter-satellite SelGH for Baseline, and with the corresponding heuristic over each satellite's observation affectation after MILP solving for the other algorithms, (ii) after intra-satellite LNS, (iii) after inter-satellite LNS. This gap is computed with reference to the best known reward for each instance. There are several results to point out. First, LKH-based algorithms perform better than ESF-based ones. Unfortunately, computing an initial schedule that respects the order obtained in the MILP solution gives worse results in most cases, since the initial solution reward computed with TourGH is lower than with SelGH, though it is less marked with LKH. A weakness in the algorithm architecture is that it requires intra-satellite scheduling to be competitive with a greedy heuristic that provides a solution very quickly, as shown in Figure 4. However, the overall performance of our LKH-based algorithms is very satisfying: after opportunistic post-insertion of non selected observations and inter-satellite scheduling, LKH-SelGH and LKH-TourGH beat the baseline algorithm when using the same amount of computation time. Our results also show that the instance archetype only slightly matters, as shown in Table 2.

Moveover, in terms of solution quality, LKH-SelGH and LKH-TourGH are competitive with the baseline algorithm running during 20 minutes. In our algorithms, the SOP solving takes several seconds per satellite. However, both LKH-SelGH and LKH-TourGH are able to outperform the baseline once in the inter-satellite phase.

Instance type	dense	sparse	mixed
Baseline (20min)	0.870	0.784	0.792
Baseline (1min)	1.05	1.53	1.15
LKH-SelGH	0.654	1.16	0.620
LKH-TourGH	0.685	1.15	0.485

Table 2: Median value of algorithms' final result depending on the instance type. The best value per instance type is highlighted in gray.

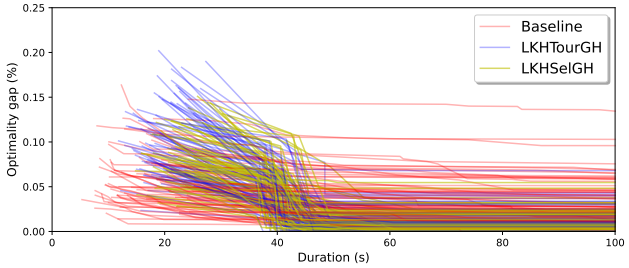


Figure 4: Optimality gaps on each instance depending on CPU time for Baseline and LKH-based algorithms. One line corresponds to a single instance.

8 CONCLUSION

This paper proposes a novel approach combining matheuristics and metaheuristics for solving EOS constellation scheduling problems. The main idea is to heuristically compute, for each satellite, a route visiting visible POIs, based on the resolution of SOPs and a multi-satellite MILP that makes some problem approximations. After that, we reconstruct actually feasible schedules for the satellites, based on some called intra-satellite and inter-satellite moves. Through schedule destroy and repair operations, the algorithm obtained is able to outperform an efficient LNS algorithm in the same amount of time.

One of the bottlenecks remains the quality of the solutions produced by the MILP and the impact of some problem approximations. Indeed, our results show that the method should not be too committed to follow the pre-determined satellite-observation assignments and observation orders. One of the next steps is to build a version of our current MILP where transition costs would be artificially increased, and then to solve the initial problem while keeping the previous satellite-observation assignment decisions. The aim would be to build a more robust assignment algorithm. In this direction, we could try and learn the parameters of such an over-constrained model to get the best results. We also believe that a Constraint Programming approach would be relevant in replacement of the matheuristic part, and we would consider comparing these different approaches.

REFERENCES

[1] Youcef Amarouche, Rym Nesrine Guibadj, Elhadja Chaalal, and Aziz Moukrim. 2020. Effective neighborhood search with optimal splitting and adaptive memory for the team orienteering problem with time windows. *Computers & Operations Research* 123 (2020), 105039. https://doi.org/10.1016/j.cor.2020.105039

[2] Valentin Antuori, Damien T. Wojtowicz, and Emmanuel Hebrard. 2025. Solving the Agile Earth Observation Satellite Scheduling Problem with CP and Local Search. In *31st International Conference on Principles and Practice of Constraint Programming (CP 2025) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 340)*, Maria Garcia de la Banda (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 3:1–3:22. https://doi.org/10.4230/LIPIcs.CP.2025.3

[3] Romain Barrault, Cédric Pralet, Gauthier Picard, and Eric Sawyer. 2025. Hybridizing Machine Learning and Optimization for Planning Satellite Observations. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 22nd International Conference, CPAIOR 2025, Melbourne, VIC, Australia, November 10–13, 2025, Proceedings, Part I* (Melbourne, VIC, Australia). Springer-Verlag, Berlin, Heidelberg, 68–85. https://doi.org/10.1007/978-3-031-95973-8_5

[4] Marco Antonio Boschetti and Vittorio Maniezzo. 2022. Matheuristics: using mathematics for heuristic design. *4OR* 20, 2 (2022), 173–208.

[5] Hermann Bouly, Duc-Cuong Dang, and Aziz Moukrim. 2008. A Memetic Algorithm for the Team Orienteering Problem. In *Applications of Evolutionary Computing*, Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Di Caro, Rolf Drechsler, Anikó Ekárt, Anna Isabel Esparcia-Alcázar, Muddassar Farooq, Andreas Fink, Jon McCormack, Michael O'Neill, Juan Romero, Franz Rothlauf, Giovanni Squillero, A. Şima Uyar, and Shengxiang Yang (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 649–658.

[6] Diego Cattaruzza, Nabil Absi, and Dominique Feillet. 2016. The Multi-Trip Vehicle Routing Problem with Time Windows and Release Dates. *Transportation Science* 50, 2 (2016), 676–693.

[7] Xiaoyu Chen, Gerhard Reinelt, Guangming Dai, and Andreas Spitz. 2019. A mixed integer linear programming model for multi-satellite scheduling. *European Journal of Operational Research* 275, 2 (2019), 694–707. https://doi.org/10.1016/j.ejor.2018.11.058

[8] W. Dullaert, G.K. Janssens, K. Sirensen, and B. Vernimmen. 2002. New heuristics for the fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society* 53, 11 (2002), 1232 – 1238. https://doi.org/10.1057/palgrave.jors.2601422 Cited by: 49.

[9] Romain Fontaine, Jilles Dibangoye, and Christine Solnon. 2023. Exact and anytime approach for solving the time dependent traveling salesman problem with time windows. *European Journal of Operational Research* 311, 3 (2023), 833–844. https://doi.org/10.1016/j.ejor.2023.06.001

[10] Ander Garcia, Pieter Vansteenwegen, Olatz Arbelaitz, Wouter Souffriau, and Maria Teresa Linaza. 2013. Integrating public transportation in personalised electronic tourist guides. *Computers & Operations Research* 40, 3 (2013), 758–774. https://doi.org/10.1016/j.cor.2011.03.020 Transport Scheduling.

[11] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. 2014. Efficient Heuristics for the Time Dependent Team Orienteering Problem with Time Windows. In *Applied Algorithms*, Prosenjit Gupta and Christos Zaroliagis (Eds.). Springer International Publishing, Cham, 152–163.

[12] Al Globus, James Crawford, Jason Lohn, and Anna Pryor. 2004. A comparison of techniques for scheduling earth observing satellites. In *Proceedings of the 16th Conference on Innovative Applications of Artifical Intelligence* (San Jose, California) *(IAAI'04)*. AAAI Press, 836–843.

[13] Keld Helsgaun. 2017. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. https://doi.org/10.13140/RG.2.2.25569.40807

[14] M. Khodadadian, A. Divsalar, C. Verbeeck, A. Gunawan, and P. Vansteenwegen. 2022. Time dependent orienteering problem with time windows and service time dependent profits. *Computers & Operations Research* 143 (2022), 105794. https://doi.org/10.1016/j.cor.2022.105794

[15] Minkeon Lee, Seunghyeon Yu, Kybeom Kwon, Myungshin Lee, Junghyun Lee, and Heungseob Kim. 2024. Mixed-Integer Linear Programming Model for Scheduling Missions and Communications of Multiple Satellites. *Aerospace* 11, 1 (2024). https://doi.org/10.3390/aerospace11010083

[16] Bohua Li, Ming Chen, Lining Xing, Yingguo Chen, and Yingwu Chen. 2025. Optimizing time-dependent multi-agile Earth observation satellite scheduling problem using deep Q-learning and ensemble heuristics. *Information Sciences* 712 (2025), 122140. https://doi.org/10.1016/j.ins.2025.122140

[17] David Pisinger and Stefan Ropke. 2019. *Large Neighborhood Search*. Springer International Publishing, Cham, 99–127. https://doi.org/10.1007/978-3-319-91086-4_4

[18] Cédric Pralet. 2023. Iterated Maximum Large Neighborhood Search for the Traveling Salesman Problem with Time Windows and its Time-dependent Version. *Computers & Operations Research* 150 (2023), 106078. https://doi.org/10.1016/j.cor.2022.106078

[19] Lili Ren, Xin Ning, and Zheng Wang. 2022. A competitive Markov decision process model and a recursive reinforcement-learning algorithm for fairness scheduling of agile satellites. *Computers and Industrial Engineering* 169 (2022). https://doi.org/10.1016/j.cie.2022.108242 Cited by: 28.

[20] Vincent F. Yu, Parida Jewpanya, Shih-Wei Lin, and A.A.N. Perwira Redi. 2019. Team orienteering problem with time windows and time-dependent scores. *Computers & Industrial Engineering* 127 (2019), 213–224. https://doi.org/10.1016/j.cie.2018.11.044

Autonomous Federation of Earth Observation Constellations: An End-to-End Demonstrator based on the DOMINO Architecture

Cyrille De Lussy
Airbus Defence and Space
Toulouse, France
cyrille.de-lussy@airbus.com

Stéphane Derrien
Capgemini
Toulouse, France
stephane.a.derrien@capgemini.com

Marie Devant
Capgemini
Toulouse, France
marie.devant@capgemini.com

Jean-Loup Farges
DTIS, ONERA, Université de Toulouse
Toulouse, France
jean-loup.farges@onera.fr

Jonathan Guerra
Airbus Defence and Space
Toulouse, France
jonathan.guerra@airbus.com

Philippe Pavero
Airbus Defence and Space
Toulouse, France
philippe.pavero@airbus.com

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

Cédric Pralet
DTIS, ONERA, Université de Toulouse
Toulouse, France
cedric.pralet@onera.fr

Jakub Rezler
ITTI
Poznan, Poland
jakub.rezler@itti.com.pl

Raivis Skadiņš
Tilde
Riga, Latvia
Raivis.Skadins@Tilde.lv

ABSTRACT

As Earth Observation (EO) constellations proliferate, interoperability and resource optimization become critical challenges. This paper introduces the integration and end-to-end validation of the DOMINO-E federation layer agents, demonstrating how their orchestration enables autonomous Earth Observation. We detail the design and validation of its three core functional units: the Virtual Assistant Service (VAS) for natural language request definition; the Coverage Service (CS) for autonomous multi-mission task dispatching and dynamic re-dispatching; and the Satellite Communication and Resource Management Service (SCRMS) for optimized ground segment allocation. Validated via an End-to-End cloud-based testbed and a Sichuan flooding scenario involving the CO3D and Pléiades Neo constellations, the framework demonstrates a significant increase in operational responsiveness. Results confirm that the DOMINO-E Federative architecture successfully automates cross-constellation optimization, reducing area waste and ensuring seamless communication planning in complex EO missions.

KEYWORDS

Satellite Constellations, Federated Systems, Earth Observation, Autonomous Scheduling, DOMINO-E, Mission Management

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

ACM Reference Format:
Cyrille De Lussy, Stéphane Derrien, Marie Devant, Jean-Loup Farges, Jonathan Guerra, Philippe Pavero, Gauthier Picard, Cédric Pralet, Jakub Rezler, and Raivis Skadiņš. 2026. Autonomous Federation of Earth Observation Constellations: An End-to-End Demonstrator based on the DOMINO Architecture. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 9 pages.

1 INTRODUCTION

Advancements in satellite performance-specifically the blurring of resolution boundaries between commercial and defense satellites-combined with New Space initiatives from privately funded actors, have transformed Earth Observation (EO) markets. This has led to worldwide competition impacting service providers, satellite manufacturers, and governmental users [4]. A shift toward smaller satellites and larger constellations has created new business opportunities, while the enhanced EO capabilities of these new constellations enable viable and sustainable ways to fulfill societal expectations for the global population [8]. However, despite the sophistication of present systems, significant room for innovation remains, particularly in the domain of Artificial Intelligence (AI).

The EO value chain is generally segmented into three categories [17]: (1) Upstream: Space and ground segment production and launches; (2) Midstream: Satellite operations, data processing, and archiving; (3) Downstream: Transforming data into value-added products and distributing them to end-users. Within the midstream segment, offline planning and scheduling-which consists in finding methods to schedule observation and upload/download tasks over a constellation-remain identified AI challenges [13]. An attractive

framework for addressing this challenge is the DOMINO architecture, a modular, service-oriented ground segment architecture designed to include multiple constellations [9].

Using the DOMINO architecture, a multi-agent federation layer, called DOMINO-E, has been developed to coordinate systems composed of independent EO missions. The goal of this federation is to allow clients requesting acquisitions of large areas to seamlessly access several satellite constellations and communication sites. This enables the composition and download of acquisitions in significantly reduced time compared to conventional, uncoordinated requests [5]. While previous research has studied the individual agents composing this federation layer in isolation [2, 3, 10–12, 14, 16, 18, 19], the specific contribution of this work is the design and validation of the *integration* of these agents within the federation layer. While pioneering works such as NASA's EO-1 SensorWeb [1] established the feasibility of autonomous science-event triggers (e.g., wildfire detection), they primarily focused on vertical integration between specific assets. In contrast, DOMINO-E addresses the challenge of horizontal federation across heterogeneous constellations. Furthermore, whereas the NASA New Observing Strategies Testbed (NOS-T) [6] provides a framework for evaluating such dynamic responses, DOMINO-E contributes a specific end-to-end architecture that combines natural-language intent (VAS) with cross-mission ground resource optimization (SCRMS), moving beyond event-triggers toward a user-centric service model.

Section 2 of this paper presents the design of the federation layer interacting with external agents, such as the mission centers of individual missions and Ground Station as a Service (GSaaS) provider systems. Sections 3, 4, and 5 detail the specific agents of the federation layer: respectively, the Virtual Assistant Service (VAS), the Coverage Service (CS), and the Satellite Communication and Resource Management Service (SCRMS). The integration of these agents into a testbed is presented in Section 6. Section 6.3 describes the demonstration scenario and its outcomes. Finally, concluding remarks are provided in Section 8.

2 ARCHITECTURAL APPROACH

This section presents the design and functional logic of the three primary services composing the federation layer. We describe the Virtual Assistant Service (VAS) for request management, the Coverage Service (CS) for task orchestration, and the Satellite Communication and Resource Management Service (SCRMS) for ground segment optimization.

The DOMINO architecture stems from the objective of making main interfaces within EO ground systems public and standardized [9]. The building blocks of the architecture, termed "dominoes", are standalone agents relying on their own infrastructure to deliver services, potentially across multiple missions. Each domino and its interfaces are natively designed for virtualized environments, specifically cloud-based infrastructures. Dominoes are categorized according to their functional roles: (i) Reactivity and monitoring; (ii) Mission and satellite command and control; (iii) Data management and processing; (iv) Multi-mission user and resource management. These modular units are combined to form a complete, flexible EO ground segment.

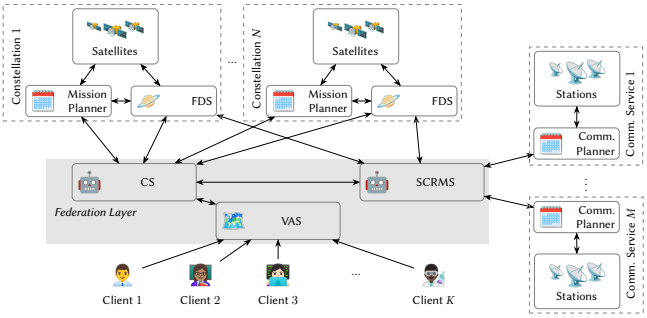


Figure 1: The DOMINO-E Federation Layer and its interactions within the ground segment architecture.

The federation layer, called DOMINO-E, addresses a complex multi-mission decision problem: how to partition a large area and assign these subdivisions to diverse missions to minimize acquisition time, ensure sufficient communication capacity, and maximize image quality-all while operating under uncertainty regarding future mission workloads. This decision process is triggered by new user requests or updates in acquisition statuses from the missions. The ecosystem involves several stakeholders: *clients* who request large-area imagery; *independent missions* (constellations) with their own planning and Flight Dynamic Service (FDS); and *communication services* (mission-owned stations or GSaaS). The global objective is to orchestrate these resources to fulfill client requests faster than uncoordinated, siloed operations [5].

Figure 1 illustrates the proposed federation layer. Fully compliant with the DOMINO architecture, this layer integrates three specific dominoes:

- **VAS (Virtual Assistant Service):** Dedicated to the management and interpretation of user requests.
- **CS (Coverage Service):** Dedicated to the orchestration and dispatching of observation tasks.
- **SCRMS (Satellite Communication and Resource Management Service):** Dedicated to the dispatching and booking of communication resources.

This federation layer acts as a seamless interface between clients and the technical complexities of multiple satellite constellations. By dividing large areas into sub-tasks that can be executed in parallel by distinct missions, the federation optimizes resource utilization and reduces the global load through the merging of overlapping requests.

The standard operational workflow of the federation layer is as follows:

(1) **Orbital Awareness:** The CS and SCRMS dominoes regularly poll FDS dominoes for up-to-date orbital parameters.
(2) **Request Interpretation:** Upon receiving a client request, the VAS interprets the natural language or structured input, derives technical observation requirements, and forwards them to the CS.
(3) **Observation Dispatching:** The CS coordinates with the individual Mission Planner dominoes to establish an agreed-upon global observation plan. This may involve iterative

negotiations or direct dispatching based on mission availability.

(4) **Communication Needs:** Once the observation plan is stabilized, the CS updates the operational needs for each satellite and requests corresponding contact slots from the SCRMS.
(5) **Contact Planning:** On a daily basis (or upon a significant update), the SCRMS computes an optimized contact plan.
(6) **Resource Booking:** The SCRMS submits booking requests to the relevant communication services (GSaaS or mission-owned).
(7) **Closing the Loop:** Upon successful booking, the SCRMS confirms the availability to the CS and, where applicable, the relevant Mission Planners, allowing for the finalization of the satellite's activity sequences.

3 VIRTUAL ASSISTANT SERVICE (VAS)

The Virtual Assistant Service (VAS) serves as the primary entry point for users within the DOMINO-E federation, providing a natural language interface to manage complex Earth Observation (EO) tasks. Unlike traditional forms or rigid command-line interfaces, the VAS allows users to define geographical areas, search product catalogs, and build multi-mission programming requests through intuitive dialogue. The VAS is justified even for expert users by its ability to synthesize cross-mission constraints into a single natural language query, reducing the need to manually navigate multiple service interfaces.

3.1 System Architecture and Base VA

The VAS is built upon a modular platform where all bots are derived from a **Base VA** framework. This framework provides standardized services for message exchange, dialogue management, and user authentication. The architecture is designed to be provider-agnostic, supporting integration with various communication channels such as the DOMINO-E User Access Service (UAS) web interface.

The core of the system is the **Dialogue Manager**, which operates according to a predefined *conversation scenario*. This scenario is structured as a directed graph of dialogue states, where transitions are triggered by detected user intents or system events. This state-machine approach ensures that the assistant maintains context-such as keeping track of a specified "Area of Interest" (AOI) while the user selects a sensor type.

3.2 Natural Language Understanding (NLU)

The NLU subsystem is responsible for converting raw text into structured data through two primary machine learning tasks:

- **Intent Detection:** The system classifies user utterances into functional categories (e.g., *SearchCatalogue*, *RequestAcquisition*). The classification is performed by a Convolutional Neural Network (CNN) architecture. User input is first transformed into 300-dimensional word embeddings using a *fastText* model. To improve model robustness, training data is synthetically augmented using LLMs like GPT-4 to generate semantically diverse examples, which are then manually curated by domain experts.
- **Named Entity Recognition (NER):** The system identifies domain-specific entities such as coordinates, dates, mission names, and sensor constraints. The VAS utilizes a hybrid NER approach: *regex-based models* are used for structured data like dates and coordinates, while *trainable models* are employed for semantic entities identified during testing and domain-specific annotations.

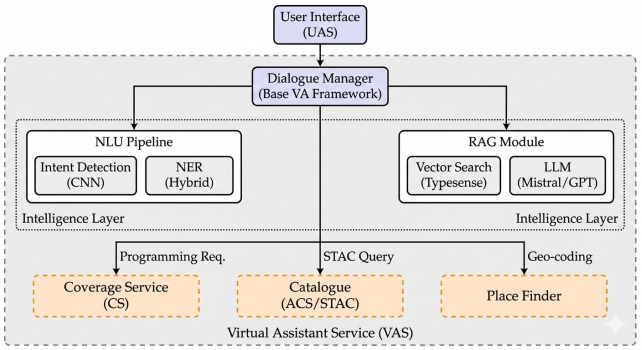


Figure 2: Functional components and integration architecture of the Virtual Assistant Service.

3.3 Hybrid Dialogue and RAG

To handle queries that fall outside the structured dialogue scenarios (e.g., "What is the resolution of Pléiades Neo?"), the VAS implements a **Retrieval-Augmented Generation (RAG)** module. This allows the assistant to answer open-ended questions using unstructured documentation stored in a local knowledge base.

The RAG process involves three stages:

(1) **Vectorization:** Knowledge base documents and user queries are converted into high-dimensional vectors using models such as Mistral or Multilingual Sentence BERT.
(2) **Retrieval:** The system retrieves the most relevant document chunks from a *Typesense* vector database based on cosine similarity.
(3) **Generation:** An LLM synthesizes a response by combining the retrieved technical context with the user's original query, ensuring the answer remains grounded in project-specific data.

3.4 Integration and External Services

The VAS acts as a technical bridge between the user and the other "dominoes" of the federation. It integrates with external systems via standardized APIs:

- **Archive/Catalogue Service (ACS):** The VAS translates user search parameters into *SpatioTemporal Asset Catalog (STAC)* API queries, allowing users to browse historical data from resources like the Copernicus Data Space Ecosystem.
- **Coverage Service (CS):** Once a user request is finalized (e.g., "Monitor the Sichuan region for 48 hours"), the VAS submits a technical *Programming Request* to the CS, which then handles the orchestration of satellite assets.
- **Toponym Search:** A Place Finder component is integrated to allow users to define geographical areas using natural names rather than just coordinates.

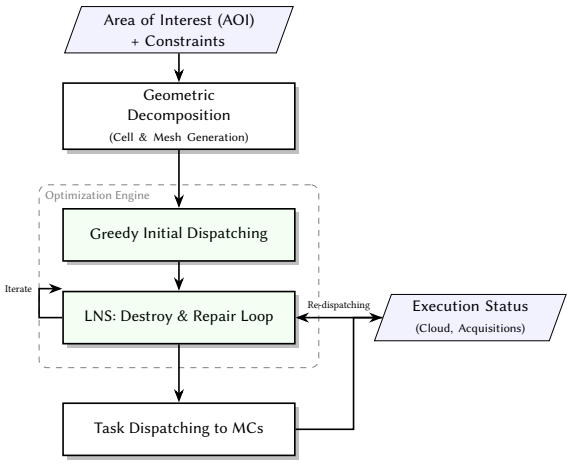


Figure 3: Coverage Service Workflow: from AOI input to autonomous re-dispatching based on mission feedback.

4 COVERAGE SERVICE (CS)

The Coverage Service (CS) is the orchestration core of the DOMINO-E federation layer. It is responsible for partitioning large geographical requests into manageable tasks and distributing them across available satellite missions. Several dispatch strategies are available in the CS. One of them addresses the "Mesh Dispatching" problem, optimizing the use of heterogeneous constellations to minimize acquisition time while accounting for system-specific constraints and execution uncertainties, following the workflow illustrated in Figure 3.

4.1 Spatial Decomposition: Meshes and Cells

To handle heterogeneous systems, the CS employs a dual-layer spatial decomposition approach:

- **Meshes:** Each mission s defines its own set of meshes M_s, aligned with a "world layer split" (typically North-South/East-West). Mesh sizes are dictated by the sensor's swath width and resolution.
- **Cells:** To ensure full coverage without gaps across varying mesh grids, the CS computes an atomic decomposition of the Area of Interest (AOI) into *cells*. A cell is defined as a maximum polygon where all internal points belong to exactly the same mesh for every available mission.

This cell-based approach provides a mathematical guarantee that covering all assigned cells leads to a complete observation of the requested area, regardless of the differing mesh geometries used by the federated missions.

4.2 Optimization Objectives

The dispatching problem is modeled using a coarse-grain representation of satellite capacities. Satellite passes are divided into temporal *slots*, each capable of observing a maximum number of meshes based on the satellite's agility. The CS optimizes a lexicographic objective function:

(1) **Minimize Completion Time (T_{max}):** Ensuring the last required image is acquired as early as possible.

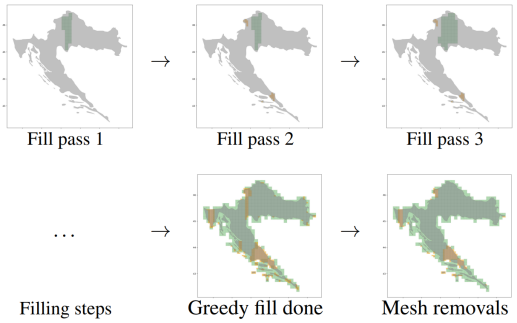


Figure 4: Iterations of the greedy search procedure selecting meshes step-by-step to cover the area of interest, for a scenario involving two systems (system 1 using the green meshes and system 2 using the yellow meshes).

(2) **Maximize Mesh Grouping:** Reducing satellite maneuver overhead by preferring contiguous "strips" of meshes within a single pass.
(3) **Minimize Area Wastage:** Avoiding overlaps between different missions and redundant observations outside the AOI.

4.3 Large Neighborhood Search (LNS) Algorithm

Given that the dispatching problem is NP-hard [14], the CS utilizes a **Large Neighborhood Search (LNS)** metaheuristic to explore the solution space efficiently.

Initial Heuristic Search. The process begins with a *greedy heuristic* that builds an initial plan. It iterates through available satellite passes chronologically, selecting meshes based on a scoring system that balances geographical utility (cells covered) and the potential for grouping with already selected meshes in the same orbit.

Destroy and Repair Cycles. The LNS then refines this initial plan through iterative cycles:

- **Destroy:** A significant portion of the current assignment (e.g., 15–30% of meshes) is removed. The algorithm employs targeted removal of meshes at the "frontiers" of mission assignments to allow for better cross-mission optimization.
- **Repair:** The removed meshes are re-inserted using the greedy heuristic, allowing the system to escape local optima and find more efficient task distributions.

The steps are illustrated in Figure 4.

4.4 Dynamic Re-dispatching and Uncertainty

A key innovation of the CS is its **autonomous re-dispatching mechanism**. Because the federation's model is a coarse approximation and factors like cloud cover or mission-level planning changes can invalidate acquisitions, the CS regularly re-optimizes the plan. During each re-dispatching cycle (e.g., once every 24 hours), the CS integrates feedback from the Mission Centers regarding successful acquisitions. It removes successfully covered cells from the problem and re-runs the LNS for the remaining AOI, potentially re-assigning tasks from a lagging mission to one with better upcoming availability or weather conditions.

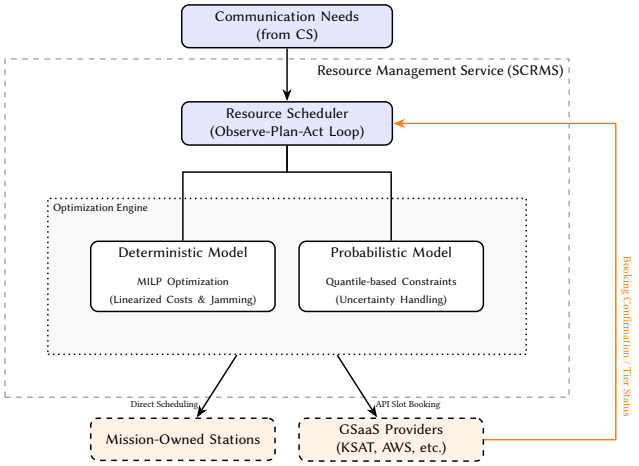


Figure 5: SCRMS Architecture: Integrating deterministic MILP and probabilistic quantile-based models for hybrid ground station scheduling.

5 SATELLITE COMMUNICATION AND RESOURCE MANAGEMENT SERVICE (SCRMS)

The SCRMS optimizes the ground segment resources for the federation, managing the scheduling and booking of satellite-to-ground contacts across both mission-owned and third-party Ground Station as a Service (GSaaS) providers.

5.1 Operational Context

The service's objective is to satisfy the communication needs of each satellite-defined by frequency bands and contact durations-while minimizing costs and avoiding interference.

Needs are categorized as *general* or *localized* (area-specific), specifying the required visibility windows for S-band (uplink/downlink) and X-band (high-speed data downlink). The optimization must account for: (i) **Cost Models:** agreements include *pay-per-pass*, *pay-per-duration*, and *commitment-based* tiers where a volume of contacts is prepaid and extra bookings incur fixed costs; (ii) **Conflict and Jamming:** Conflicts arise when two satellites attempt to access the same station/band simultaneously. Jamming occurs between nearby stations on the same site based on the angular separation of the station-satellite vectors.

The SCRMS operates on an *Observe-Plan-Act* loop [15], triggered daily or by emergency updates:

(1) **Observe:** Gathers communication needs and computes potential visibility windows using the Orekit space dynamics library [7].
(2) **Plan:** Solves 10 independent daily multi-criteria optimization problems over a 10-day horizon, ensuring fulfillment of needs while minimizing costs and interference.
(3) **Act:** Manages the asynchronous booking of contacts through GSaaS APIs, respecting tier-based booking windows.

5.2 Optimization Modeling

The problem involves N satellites, each with L_i potential contacts and K_i needs. Interference between contact l of satellite i and contact m of satellite j is represented by parameter $b_{i,l,j,m}$. The full description of these models are available in [18].

Deterministic Approach. The selection is modeled using binary variables $x_{i,l} \in \{0, 1\}$. While the cost for commitment-based agreements and the jamming criterion J_{in} are naturally non-linear, the problem is linearized to employ Mixed-Integer Linear Programming (MILP) via the Google OR-Tools solver. The non-linear interference term $\sum b_{i,l,j,m} x_{i,l} x_{j,m}$ is linearized using auxiliary variables $y_{i,l,j,m}$ and standard constraints ($y \le x_{i,l}, y \le x_{j,m}, y \ge x_{i,l} + x_{j,m} - 1$). Similarly, commitment costs are linearized by introducing variables to represent the volume of contacts above the prepaid threshold Y_s. Experiments show that the MILP approach significantly outperforms round-robin or greedy heuristics in both solution quality and computation time [19].

Probabilistic Approach. To account for the uncertainty of GSaaS contact acceptance, we introduce a probability of success $p_{i,l}$. Simply optimizing expected values would result in failing to satisfy approximately half of the requests. Consequently, we incorporate the standard deviation of need fulfillment into the constraints:

$$\sum_l d_{i,k,l} p_{i,l} x_{i,l} - q\left(\sum_l d_{i,k,l}^2 p_{i,l} x_{i,l} (1 - p_{i,l})\right)^{\frac{1}{2}} \ge D_{i,k}$$

where $D_{i,k}$ is the required time for need k and q controls the fulfillment quantile. By linearizing this standard deviation term, the problem remains solvable via MILP. Numerical results with $q = 2$ demonstrate that this probabilistic approach reduces the communication time deficit from 10% in the deterministic model to 6% [18].

6 TEST-BED INTEGRATION AND DEMONSTRATION

The operational capabilities of the DOMINO-E architecture were validated through a comprehensive End-to-End demonstrator. The demonstration utilizes a shared cloud testbed where the Virtual Assistant Service (VAS) operates in a dedicated namespace, while the Coverage Service (CS) and the Satellite Communication and Resource Management Service (SCRMS) are deployed together to ensure low-latency orchestration.

6.1 Test-bed Architecture

As part of the DOMINO-E initiative, all the developed dominoes have been integrated together. A common platform, called the Test-bed, has been created for this purpose (see Figure 6). Hosted on a Google Cloud Platform located in Europe, the biggest challenge of this platform building was to manage high security standards and multinational accesses.

Once provisioned, this platform has been used by all the dominoes, for their own validation, and for a common validation within the framework of an End-to-End test, involving all the dominoes.

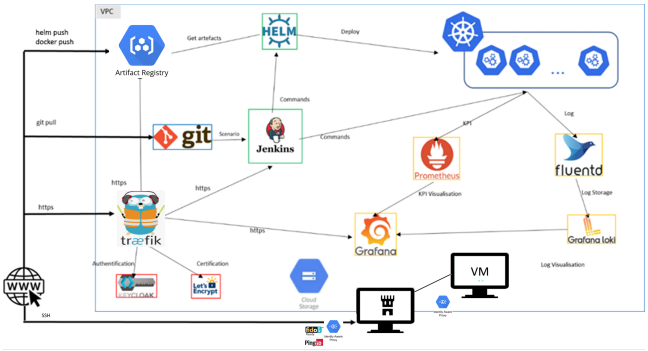


Figure 6: Test-bed architecture and technologies

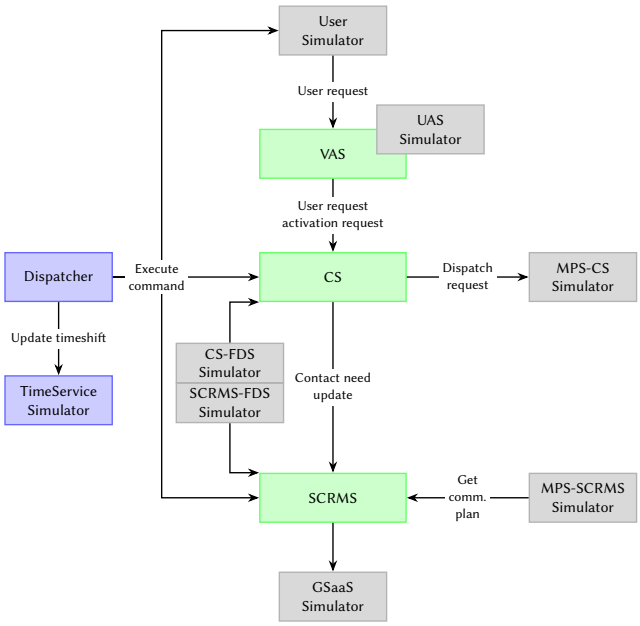


Figure 7: Interactions within the test-bed: dominoes in green, Simulators in grey, and Dispatcher/Scheduler in blue.

6.2 End-to-End Scenario

The underlying story behind the End-to-End scenario is to react to a large scale catastrophic event such as a flood or a hurricane (in the region of Sichuan in our example), by managing the deposit and the planing of imagery request of this area. The End-to-End scenario has been build to trigger interactions between the Virtual Assistant, the Coverage Service and the SCRMS: The User request deposit goes through User Access Service using the Virtual Assistant Service, is dispatched to the constellations by the Coverage Service. This implies a change in download needs, which triggers the SCRMS to change the communication plan. All these steps are displayed in Figure 7, where the inter-component interaction involves a closed-loop feedback from VAS to CS.

The End-to-End validation of the DOMINO-E federation layer was conducted using a high-fidelity simulation environment designed to mimic real-world space operations. The simulation environment provides an integrated testbed where the various "Domino" services (VAS, CS, and SCRMS) are deployed within shared cloud namespaces. To represent the space segment, the simulation incorporates digital twins of the CO3D and Pléiades Neo constellations, allowing for the modeling of realistic orbital dynamics and instrument constraints (FDS), and of the mission planning centers (MPS). The ground segment is simulated through a federated network model that includes both mission-owned stations and virtualized GSaaS (Ground Station as a Service) providers, such as KSAT. Furthermore, a User Simulator was utilized to perform stress tests and rapid-fire validation runs, bypassing the manual chat interface to evaluate the system's algorithmic performance under high-frequency request loads.

6.3 Demonstration

The End-to-End demonstration is available as a video, here: https://www.youtube.com/watch?v=I2biwBIydZ0.

User Interaction and Request Definition. The workflow begins at the **User Access Service (UAS)** interface, where the user initiates a request via the natural language chat interface. In the demonstration scenario, the user defines an observation area over **Sichuan, China**, a region frequently prone to flooding. The VAS guides the user through defining the period of interest and fine-tuning mission parameters for two federated constellations: **CO3D** and **Pléiades Neo**.

Autonomous Coverage Planning and Dispatching. Once the user validates the request, the VAS transfers the technical parameters to the CS. The system's autonomous planning involves several key steps:

- **Decomposition:** The CS divides the target area into multiple independent programming requests and displays them on a global map for user follow-up.
- **Initial Dispatching:** Tasks are distributed between the available constellations, and a progress graph provides an estimated completion timeline.
- **Dynamic Re-dispatching:** The demonstrator showcases a scenario where the system automatically triggers a re-dispatch. For instance, tasks originally assigned to Pléiades Neo are canceled and resubmitted to CO3D to optimize the total completion time based on real-time mission progress.

Communication and Resource Optimization. Following the re-dispatch of observation tasks, the CS communicates updated contact needs to the **SCRMS**. The SCRMS automatically aligns the communication plan with the new coverage requirements, adjusting the number of booked ground station slots to ensure efficient data downlink.

Validation Results. The final state of the demonstration confirms the stability of the federation layer. The results show that: (i) The system achieved approximately 20% total coverage within the simulated window, successfully handling task cancelations and re-assignments. (ii) The SCRMS maintained a close alignment between

"required" and "booked" contact durations, confirming that the communication plan remained optimized even as the observation plan evolved. All operations, from initial request to final downlink booking, are executed automatically, demonstrating the high level of autonomy achieved by the DOMINO-E federation layer.

7 OUTCOMES

This section details the performance assessment of the individual services within the DOMINO-E federation. The evaluation focuses on operational efficiency, algorithmic accuracy, and resource consumption. We consider two operational scenarios. The *Occitania* scenario covers a 72,000 km² region in southern France over a 48-hour horizon, while the *France* scenario involves a nationwide coverage request to test scalability. All experiments were conducted on a Linux-based server with a 3.6 GHz Intel Core i7 processor and 32GB RAM.

7.1 Virtual Assistant Service (VAS) Outcomes

The performance of the VAS was evaluated based on its ability to streamline user interactions compared to traditional Graphical User Interfaces (GUIs). Assessment metrics focused on task completion time, dialogue accuracy, and system robustness.

Operational Efficiency. The VAS demonstrated a significant reduction in the time required to perform core Earth Observation (EO) tasks. Comparative tests against the standard User Access Service (UAS) manual interface revealed the following improvements:

- **Catalogue Search:** Utilizing the VAS for area-of-interest selection reduced search times from 131 seconds to 83 seconds on average.
- **Programming Request (PR) Creation:** For expert users, the creation of PRs was reduced from a baseline of 247–340 seconds (UAS manual) to 102–211 seconds when assisted by the VAS.
- **High-Speed Processing:** Automated user simulations achieved task completion in as little as 28 seconds, highlighting the efficiency of the NLU-driven backend.

Dialogue Quality and LLM Reasoning. The hybrid NLU architecture, combining structured dialogue management with LLM-based reasoning, was assessed for response relevance and accuracy.

- **Response Accuracy:** 64.7% of responses were classified as fully correct and context-aware.
- **Advanced Reasoning:** 20.2% of interactions were rated as "superior," where the assistant successfully managed complex follow-up questions, relative date handling (e.g., "two weeks after"), and parameter disambiguation.
- **Failure Modes:** Approximately 12.1% of responses were inadequate, primarily occurring in excessively long dialogue chains where context retention reached its limit.

The 64.7% accuracy primarily stems from NLU intent classification errors in complex multi-parameter queries; however, the 'Human-in-the-loop' design allows users to correct these parameters before submission.

Table 1: Key Performance Indicators (KPIs) for the VAS

Metric	Manual Interface	VAS Assisted
Avg. Image Search Time	131 s	83 s
Avg. PR Creation Time	293 s	157 s
System Accuracy	N/A	84.9%
Memory Footprint (per instance)	N/A	~165 MB

Resource Consumption and Stability. Monitoring of the VAS during peak loads confirmed the service's stability. In concurrent automated testing scenarios, memory consumption for the core processing node remained stable at approximately 165 MB. The system showed high resilience to non-ideal inputs, including spelling errors and incomplete coordinate formats.

The evaluation confirms that the VAS effectively lowers the barrier to entry for complex multi-mission programming, enabling intuitive access to the federation's underlying optimization services.

7.2 Coverage Service (CS) Outcomes

The evaluation of the CS centers on "Area Waste" (the ratio of redundant or unnecessary acquisition area to the target area) and "Completion Time" across different algorithmic use cases: manual dispatch (UC1a), static dispatch (UC1b), dynamic re-dispatch with planning estimation (UC1c), and Large Neighborhood Search (UC2).

Operational Efficiency and Completion. The CS demonstrated the capability to coordinate multiple constellations to achieve high completion rates for large-scale requests.

- **Scenario Performance:** In the medium-scale *Occitania* scenario, the dynamic re-dispatching (UC1c) achieved a 54.63% completion after 10 days of simulated operations.
- **Scalability:** The service successfully handled the *France* scenario (hundreds of thousands of km²), managing the increase in dispatched programming requests proportionally to the area size.
- **Redispatching Efficiency:** Dynamic redispatching was shown to adapt to real planning progression, reducing the estimated time to full completion by several days compared to static methods (e.g., reaching full coverage estimates by 04/07/2034 instead of 08/07/2034 in the Occitania test starting on 01/06/2034).

Algorithmic Accuracy and Area Waste. A key performance indicator for the CS is the minimization of "Area Waste" during the mesh decomposition and dispatch phase.

- **Dispatch Precision:** The Large Neighborhood Search (UC2) significantly outperformed other methods in terms of precision, taking the actual splitting of mission chains as input. This resulted in an **Area Waste of 20.08%** for the loaded Occitania scenario, compared to over 50% for standard decomposition methods (UC1b/c).
- **Mission Optimization:** When taking mission chain self-optimization into account, the waste for UC1c dropped from 50.78% to a more manageable 22.31%, validating the benefit of federated feedback.

Computational Performance. The CS maintains efficient computation times, even when managing complex redispatching logic.

Table 2: Key Performance Indicators (KPIs) for the CS

Metric	Static Dispatch	LNS Dispatch
Coverage Completion (10 days)	32.6%	26.46%
Raw Area Waste	50.16%	20.08%
Waste after Mission Chain Opt.	21.99%	15.43%
Redispatch Time (s)	N/A	51.85

Table 3: Key Performance Indicators (KPIs) for the SCRMS

Metric	Owned Stations Only	Federated GSaaS
X-Band Fulfillment Rate	64.2%	100%
Avg. Data Latency	High	Low
Peak Memory Usage	~450 MB	~1 GB
CPU Cores (Peak)	2	4

- **Computation Speed:** Initial dispatch for medium areas averaged approximately 70.65 seconds.
- **Dynamic Updates:** Subsequent redispatches were faster (37.38 seconds average), as they leveraged the previous state as an input and dealt with smaller remaining areas.
- **Resource Usage:** The number of external calls remained balanced, with dynamic use cases successfully managing thousands of interactions with mission chains to track status and progress.

The outcomes confirm that the CS effectively bridges the gap between high-level user requests and low-level mission scheduling, providing an autonomous layer that optimizes global coverage while minimizing resource redundancy.

7.3 Satellite Communication and Resource Management Service (SCRMS) Outcomes

The SCRMS was evaluated on its ability to satisfy multi-mission communication needs while optimizing ground station (GS) slot allocation. The assessment compared a "Minimum Configuration" (mission-owned sites only) against the "Federated Configuration" (mission-owned plus GSaaS providers).

Fulfillment of Communication Needs. The integration of a federated GS network significantly improved the system's ability to meet high-bandwidth requirements:

- **X-Band Capacity:** In the minimum configuration, owned sites were often insufficient to meet the peak data downlink demands of the CO3D and Pléiades Neo constellations. The federated approach, by booking external GSaaS slots, consistently achieved 100% fulfillment of X-band needs.
- **S-Band Reliability:** S-band needs for command and control were satisfied across all scenarios, with the scheduler successfully managing priority levels for critical telemetry links.

Resource Optimization and Cost. The SCRMS demonstrated high efficiency in resource allocation through its dual optimization logic:

- **Grouping Efficiency:** The scheduler achieved significant "Grouping" of contacts, bundling multiple acquisitions into single ground station visibility windows to reduce antenna slewing and operational overhead.
- **Cost Management:** While the minimum configuration has zero external cost, it suffers from data latency. The federated configuration utilized the cost-per-pass and commitment models to find a Pareto-optimal balance between data freshness and booking expenditure.

Computational Stability and Hardware Usage. The service maintained a modest hardware footprint even when managing complex multi-day horizons:

- **Memory Footprint:** Memory usage of the SCRMS Docker instance peaked at approximately 1 GB during the solving phase of large scenarios.
- **CPU Performance:** The solver reached a peak of 4 cores during the initial optimization of the 10-day horizon, stabilizing at 1 core for subsequent re-optimization and acting loops.
- **Acting Latency:** The automated "Act" loop, responsible for API interactions with GSaaS providers, maintained sub-second latency for slot booking confirmations.

The results validate the SCRMS as a critical enabler for the federation, transforming the "Communication Needs" generated by the Coverage Service into an executable and cost-efficient ground segment schedule.

8 CONCLUSION

The DOMINO-E project has successfully demonstrated the feasibility and operational benefits of a multi-mission federation layer for Earth Observation. By moving beyond traditional, fragmented mission management, the project has established an autonomous ecosystem-the "DOMINO" architecture-that effectively bridges the gap between complex user requirements and heterogeneous satellite constellations.

The results obtained across the three core services-VAS, CS, and SCRMS-consistently validate the project's objectives:

(i) **Accessibility:** The VAS reduced tasking time by 40% via natural language, democratizing access for non-experts.
(ii) **Efficiency:** The CS utilized LNS algorithms to reduce area waste to 20%, significantly optimizing large-scale monitoring compared to static methods.
(iii) **Scalability:** By transitioning to a GSaaS model, SCRMS achieved 100% fulfillment of X-band communication needs.

The "Closed-Loop" capabilities showcased in the End-to-End demonstration highlight the system's ability to autonomously redispatch tasks in response to real-world uncertainties. Ultimately, DOMINO-E provides a scalable framework for future European initiatives, establishing a foundation for a more responsive and efficient space segment.

ACKNOWLEDGMENTS

This work is part of the DOMINO-E project which received funding from the European Union's Horizon Europe Programme for Research and Innovation under Grant Agreement n°101082230.

REFERENCES

[1] S. Chien, B. Cichy, A. Davies, D. Tran, G. Rabideau, R. Castano, R. Sherwood, S. Nghiem, R. Greeley, T. Doggett, V. Baker, J. Dohm, F. Ip, D. Mandl, S. Frye, S. Shuman, S. Ungar, T. Brakke, L. Ong, J. Descloitres, J. Jones, S. Grosvenor, R. Wright, L. Flynn, A. Harris, R. Brakenridge, and S. Cacquard. 2006. An autonomous earth observing sensorWeb. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, Vol. 1. 8 pp.–. https://doi.org/10.1109/SUTC.2006.1636174

[2] Cyrille De Lussy and Jonathan Guerra. 2024. Domino-E Coverage Service. Poster to the 9th European Mission Operations Data System Architecture Workshop (ESAW). Darmstadt, Germany.

[3] Cyrille de Lussy, Jonathan Guerra, Vivien Nguyen, Gauthier Picard, Cédric Pralet, Filipo Studzinski Perotto, Stéphane Derrien, Matthieu Vansteene, Corentin Roux, and Jean-François Vinuesa. 2024. Domino-E coverage service - A flexible, smart automated tool for multi-mission federation. In *75th International Astronautical Congress - Earth Observation Symposium*.

[4] Gil Denis, Alain Claverie, Xavier Pasco, Jean-Pierre Darnis, Benoît de Maupeou, Murielle Lafaye, and Eric Morel. 2017. Towards disruptions in Earth observation? New Earth Observation systems and markets evolution: Possible scenarios and impacts. *Acta Astronautica* 137 (2017), 415–433.

[5] Jean-Loup Farges, Filipo Studzinski Perotto, Cédric Pralet, Gauthier Picard, Cyril de Lussy, Jonathan Guerra, Philippe Pavero, and Fabrice Planchou. 2024. Going beyond mono-mission earth observation: Using the multi-agent paradigm to federate multiple missions. In *23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS-24)*. International Foundation for Autonomous Agents and Multiagent Systems, 2674–2678.

[6] Paul T. Grogan, Hayden C. Daly, Matthew S. Brand, and Jerry J. Sellers. 2021. New Observing Strategies Testbed (NOS-T) Architecture: Evaluating Dynamic Response to Emergent Events. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. 1470–1473. https://doi.org/10.1109/IGARSS47720.2021.9555131

[7] maisonobe, Bryan Cazabonne, Evan Ward, Romain Serra, Maxime Journot, Sébastien Dinot, Guilhem Bonnefille, Thomas Neidhart, Luc Maisonobe, Clément Jonglez, Mark Rutten, yjeand, Julio Hernanz, lirw1984, Vincent Cucchietti, Andrew Goetz, Guiuux, gaetanpierre0, Lars Næsbye Christensen, Alberto Fossà, jvalet, Alberto Ferrero, gabb5, liscju, Christopher Schank, Tanner Mills, plan3d, Vyom Yadav, Shiva Iyer, and Petrus Hyvönen. 2024. *CS-SI/Orekit: 12.2*. https://doi.org/10.5281/zenodo.13950582

[8] Salvo Marcuccio, Silvia Ullo, Marco Carminati, and Olfa Kanoun. 2019. Smaller satellites, larger constellations: Trends and design issues for earth observation systems. *IEEE Aerospace and Electronic Systems Magazine* 34, 10 (2019), 50–59.

[9] Daniel Marcell Novak, Amina Annane, Régis Baillard, Alain Berry, Cédric Brandon, Vincent Desormeau, Sylvain Gaudan, Julien Joyeux, Stan Kaethler, Charlie Madier, Olivier Melet, and Yann Roux. 2022. Future ground segments with standardized interfaces: the DOMINO-X project. In *73rd International Astronautical Congress*.

[10] Philippe Pavero. 2024. Domino-E Satellite Communication and Resource Management Service. Poster to the 9th European Mission Operations Data System Architecture Workshop (ESAW). Darmstadt, Germany.

[11] Philippe Pavero, Jean-Loup Farges, Gauthier Picard, Jakub Rezler, and Jean-François Vinuesa. 2025. Satellite Communication Management Domino, for Constellation and Ground Station as a Service Interconnection. In *76th International Astronautical Congress - Earth Observation Symposium*.

[12] Philippe Pavero, Fabrice Planchou, Jean-Loup Farges, Gauthier Picard, Filipo Perotto, Jakub Rezler, and Jean-François Vinuesa. 2024. Satellite communication management Domino, for constellation and ground station as a service interconnection. In *75th International Astronautical Congress - Earth Observation Symposium*.

[13] Gauthier Picard, Clément Caron, Jean-Loup Farges, Jonathan Guerra, Cédric Pralet, and Stéphanie Roussel. 2021. Autonomous Agents and Multiagent Systems Challenges in Earth Observation Satellite Constellations. (2021).

[14] Cédric Pralet, Gauthier Picard, Cyrille de Lussy, and Jonathan Guerra. 2025. Mesh Dispatching for Area Coverage using Several Earth Observation Systems. In *IWPSS-International Workshops on Planning and Scheduling for Space*.

[15] Matthew Roberts. 2002. Artificial Actors for Real World Environments. In *2002 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*. Association for the Advancement of Artificial Intelligence, 87–94.

[16] Raivis Skadiņš, Evita Korņējeva, and Modris Stūrmanis. 2015. *D6.5 – Demonstration of the VAS*. EU project Technical Report D29. Tilde SIA.

[17] Wasanchai Vongsantivanich, David LX Ho, and Quentin Verspieren. 2018. Using space for disaster management in emerging space states: A critical assessment. (2018).

[18] Hénoïk Willot, Jean-Loup Farges, Gauthier Picard, and Philippe Pavero. 2025. Deterministic and Probabilistic Decision Models for GSaaS-based Satellite Communication Resource Management. In *IWPSS-International Workshops on Planning and Scheduling for Space*.

[19] Hénoïk Willot, Jean-Loup Farges, Gauthier Picard, and Philippe Pavero. 2025. Satellite communication resources management in a earth observation federation of constellations. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 235–251.

Effects of Lookahead and Communication Dynamics on Multi-Agent Potential-Field Rover Navigation

Timothy Flavin
University of Tulsa
Tulsa, United States
Timmy-Flavin@utulsa.edu

Joe Shymanski
University of Tulsa
Tulsa, United States
Joe-Shymanski@utulsa.edu

Sandip Sen
University of Tulsa
Tulsa, United States
Sandip-Sen@utulsa.edu

ABSTRACT

In this work, we investigate how predictive lookahead and communication impact multi-agent stochastic artificial potential field navigation (SPF) for rovers. We choose SPF to balance exploration, collision avoidance, and hazard evasion using terrain data derived from real HiRISE Mars terrain data. We empirically evaluate how varying lookahead horizons, prior belief map inaccuracies, and communication frequency impact effective spatial coverage. Our key results show that while minimal communication ($p = 0.05$) successfully bounds lookahead prediction error, naive SPF controllers struggle to use predictive information effectively. Additionally, we found that lookahead combined with environment stochasticity act as an effective local-minimum escape strategy when policy noise is not an option. Ultimately, this work provides a computationally efficient engine and first benchmark for testing multi-agent coverage on real world terrain. Our results suggest lookahead with sparse communication is sufficient for teammate modeling, but taking advantage of that information is beyond simple SPF controllers.

KEYWORDS

Stochastic Potential Field Navigation, Simulated Annealing, Multi-Rover Navigation

ACM Reference Format:
Timothy Flavin, Joe Shymanski, and Sandip Sen. 2026. Effects of Lookahead and Communication Dynamics on Multi-Agent Potential-Field Rover Navigation. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 5 pages.

1 INTRODUCTION

Future space missions increasingly envision teams of autonomous robotic assets operating in hazardous and partially known environments [6, 15, 19]. Such distributed robotic systems must operate under strict communication constraints, uncertain terrain, and limited sensing. Centralized coordination is often infeasible due to latency and bandwidth limits, making lightweight decentralized decision-making a critical enabler of scalable space exploration.

A fundamental task in such missions is spatial coverage: multiple rovers must efficiently explore and map an unknown region while avoiding hazardous terrain and minimizing redundant traversal [5, 21]. Under partial observability, agents must balance exploration and danger avoidance, anticipate teammate movements to prevent overlap, and compensate for stale or noisy information. In realistic planetary scenarios, prior terrain maps derived from orbital imagery may also contain systematic errors.

Artificial Potential Fields (APFs) provide a computationally lightweight and interpretable mechanism for decentralized navigation [12]. By superimposing attraction and repulsion forces derived from spatial features, agents compute continuous control actions without requiring large neural models. However, classical APF approaches largely assume reactive behavior [12, 17], leaving open questions about the impact of forward prediction and communication.

In this work, we investigate how lookahead simulation and communication constraints interact in multi-agent potential-field rover navigation. We model four rovers exploring a hazardous terrain grid derived from real HiRISE Digital Terrain Models [1, 14], operating under partial observability with probabilistic communication updates. Rover policies are parameterized as weighted superpositions of spatial potential fields and optimized via simulated annealing under varying communication settings. To enable teammate anticipation, agents simulate peer trajectories over configurable lookahead horizons using internal belief states [8, 13]. Controlled noise injected into prior terrain maps models discrepancies between orbital estimates and true surface hazards. This framework allows us to examine three core questions: (1) how lookahead depth affects coordination under imperfect teammate modeling, (2) how prior map inaccuracies interact with predictive coordination strategies, and (3) how communication frequency mitigates or amplifies lookahead mismatch.

Our contributions are threefold:

(1): We introduce a C++ engine for studying decentralized rover coverage under hazardous terrain and communication constraints.

(2): We analyze lookahead dynamics in potential-field navigation, identifying conditions under which prediction improves coordination and conditions in which mismatch degrades performance.

(3): We characterize the interplay between communication reliability, prior map uncertainty, and predictive teammate modeling in multi-agent spatial exploration.

These results provide insight into the design of decentralized autonomy architectures for multi-rover space missions, clarifying when reactive strategies suffice and when predictive modeling or communication becomes essential.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

2 BACKGROUND AND RELATED WORK

Our framework intersects three major pillars of multi-agent systems (MAS) research: spatial coverage through artificial potential fields, lookahead dynamics for teammate anticipation, and the effects of communication/transparency on overcoming partial observability.

2.1 Multi-Agent Coverage and Artificial Potential Fields

Spatial coverage and coordination in partially observable environments are frequently addressed using heuristic-driven navigation, most notably Artificial Potential Fields (APF). Originally developed for collision avoidance, APFs have been widely adapted for multi-agent dispersion, where agents are repelled by obstacles and peers but attracted to unobserved frontiers [12, 17]. Recent work has successfully integrated APFs into Multi-Agent Reinforcement Learning (MARL) to solve complex Coverage Path Planning (CPP) problems, using spatial graphs and curiosity-driven intrinsic rewards to minimize coverage gaps [11, 22].

2.2 Lookahead Dynamics and Teammate Anticipation

In decentralized, partially observable settings, agents often rely on internal forward models to anticipate allied movements. Predicting teammate behavior mitigates the "teammate delay" issue caused by concurrent policy updates [4]. Approaches like Higher-Order Gradients (HOG), Off-Policy Action Anticipation (OffPA2), and Policy Mirror Descent with Lookahead formally integrate lookahead mechanisms so agents can explicitly model the future states or actions of their peers [2, 18].

Our work grounds these theoretical lookahead models in a highly parallelized C++ environment, allowing agents to run exact forward-simulations of their peer's policies based on their current belief state. Crucially, we introduce systemic error into these forward models via inaccurate prior belief maps, forcing agents to navigate the discrepancy between predicted environment danger and dynamic hazard realities. This systemic error and policy noise are the only discrepancies between agent lookahead and reality when communication bandwidth is infinite.

2.3 Predictability, Transparency, and Communication Trade-offs

When communication bandwidth is constrained, agents must act transparently for their peers' lookahead models to remain accurate. Recent MARL research emphasizes *alignment-driven intrinsic rewards*, where agents are incentivized to learn behaviors that match their neighbors' expectations [16], naturally breaking coordination symmetries. Similarly, *Predictability Awareness* mechanisms encourage agents to foster soft social conventions; by minimizing the discrepancy between an internal prediction model and actual observations, agents effectively lower the uncertainty of the entire swarm [9]. Our research builds directly on this by quantifying the trade-off: when terrain realities diverge from the shared global map, agents must either rely heavily on explicit communication to resynchronize the swarm or adopt highly predictable local policies to minimize the resulting lookahead drift.

3 METHODOLOGY

3.1 Simulation Framework

We evaluate our hypotheses using our custom environment named *multi-agent-coverage*, a high-performance, batched multi-agent grid-world environment implemented in C++ with OpenMP for parallel execution. The environment simulates agents exploring a 32×32 grid world containing continuous danger/hazard mappings, h, in the range $[-1, 1]$.

Each agent may propose a movement vector $\vec{v} = [\delta x, \delta y]$ which is clamped to a maximum magnitude of 1.0 before being reduced again by the current danger in $\vec{v} = \vec{v}(1 - c \cdot \max(h, 0.0))$. The constant c determines the minimum speed. Every frame that a rover spends on a tile with $h > 0$ has a $p = ch$ probability of getting stuck. Stuck means that the agent's movement speed is zero until the environment resets.

Our C++ architecture ensures zero-copy memory sharing with PyTorch tensors, allowing for exceptionally high throughput during large-scale policy evaluation. This supports rapid rover belief updates and enables efficient instantiation of environment copies from an agent's internal state for future integration with Monte Carlo Search [3] or Prioritized Replay [20].

3.2 Stochastic Potential Field Navigation

Agent behavior is governed by a parameterized force-based policy. Agents compute an action vector through a linear combination of dynamic spatial features. The force exerted by grid cell i and the resulting weighted superposition are defined as:

$$\vec{F}_i = \frac{m_i \cdot \hat{r}_i}{\|\mathbf{r}_i\|^{p_i}} \qquad \mathbf{a}_t = \sum_{k=1}^{K} w_k \vec{F}_k \tag{1}$$

where $\hat{r}_i$ is the unit vector to cell i, $\|\mathbf{r}_i\|$ is the Euclidean distance, p_i is the distance exponent, and m_i is the magnitude. The weights $w_k \in [-1, 1]$ represent signed attraction/repulsion for $K = 7$ spatial features: (1) Region Danger Prior, (2) Observed Danger, (3) Visited Status, (4) Agent Locations, (5) Path Recency, (6) Boundaries, and (7) Voronoi Partitions, plus noise to avoid local minima.

Each of these fields, apart from 1 and 6, is subject to partial observability constraints and calculated from imperfect belief states. Self-Path recency sets recently visited cells to a magnitude of 1.0 with exponential decay over time, serving as an anti-pheromone inspired by path-finding algorithms such as Ant Colony Optimization [7]. Voronoi Partition masks unobserved tile magnitudes if they are closer to another agent, allowing a first-order estimation of task responsibility.

3.3 Policy Optimization via Simulated Annealing

To discover optimal potential field parameters, we deploy a Simulated Annealing (SA) hyperparameter search. The search optimizes a sequence of genes representing the specific hyperparameters being evaluated (specifically the weight w_k and the exponent p_i) for each spatial feature. Fitness is defined as the discounted cumulative

coverage reward across a batch of environments:

$$f = \sum_{t=0}^{T} \gamma^t \left(10\,\mathbb{I}[\text{done}_t] + o_t - 100\,s_t\right),$$

where $T = 1000$, γ is the discount factor, o_t is the number of newly observed tiles at timestep t, s_t is the number of rovers that become immobilized due to excessive hazard, and $\mathbb{I}[\text{done}_t]$ is an indicator equal to 1 if full coverage is achieved at time t. This fitness directly captures the mission's dual objectives: maximizing spatial discovery while strictly penalizing rover loss due to hazards.

4 EXPERIMENTAL DESIGN

Our experimental maps are constructed from real Digital Terrain Models (DTMs) obtained via the HiRISE mission, centered on a prospective Mars landing site. Sixteen 1km^2 terrain slices were processed into dual mappings: a maximum gradient map and an elevation standard deviation map, serving as proxies for surface steepness and roughness, respectively. To assess the intersection of transparency, lookahead, and communication, we benchmark optimized rovers across all environments on a matrix of conditions:

(1) Communication Levels: We evaluate full, partial, and zero-communication scenarios. Communication consists of [x,y] location and "is_alive" updates to other rovers.

(2) Lookahead Depth: Agents simulate peer movements for 0 to 10 time-steps (up to 300 meters traversal depending on terrain) before selecting actions. With Lookahead (LA) enabled, agents update unseen teammate locations using their internal belief state.

(3) Prior Map Noise: To simulate real-world inaccuracies, controlled Gaussian noise is injected into the expectation maps during zero-communication runs as shown in Figure (2).

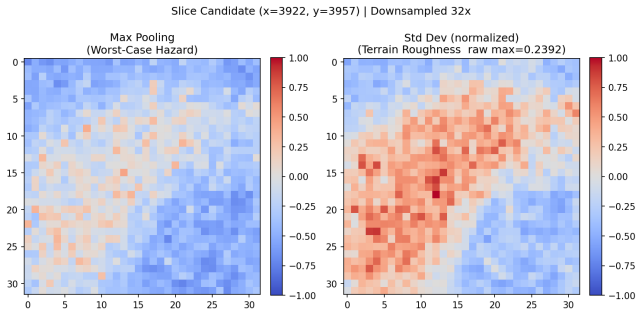


Figure 1: Example terrain slice (1024 × 1024 m). Each cell corresponds to a max-pooled 32 m^2 region derived from HiRISE DTM data.

By masking which features are available to a particular rover's SPF navigation function, we create three baseline policies trained with no communication or lookahead enabled. **Global** and **Random** have access to all variables including ground-truth and partially observed variables. **Global** starts with a noisy Voronoi policy and **Random** begins with random uniform weights over all features. We also mask out all but the local variables for **Local**, and we train variants via simulated annealing specifically for lookahead and communication $p = 0.05$: **Local+LA** and **Random+LA**.

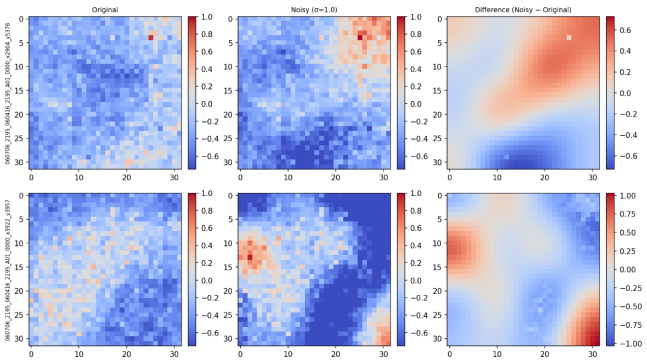


Figure 2: Two map examples with (left) original max-pooled gradient map, (center) noise-perturbed prior map, and (right) the injected Gaussian noise field.

We aim to test several hypotheses about the impact of lookahead, communication, and policy entropy on team performance:

Hypothesis 1: As policy noise weight increase, lookahead prediction will grow significantly correlating with degraded performance. Figures (3, 4).

Hypothesis 2: As the prior belief map of terrain danger becomes more noisy, lookahead will become more destructive. Figure (5).

Hypothesis 3: Lookahead accuracy will improve significantly with occasional correction from communication, improving team performance. Figure (6).

5 RESULTS

To evaluate our hypotheses, we analyzed both coverage performance and ally prediction error across varying lookahead steps and noise conditions.

Lookahead under Imperfect Teammate Modeling (Hypothesis 1): Figures 3 and 4 demonstrate that without the occasional resetting (1/20 chance) to ground truth, lookahead introduced mean squared error and reduced performance for already stochastic policies, but lookahead helped deterministic policies performance. This is because a local minimum in the potential field only changes if an agent's belief about its teammates changes.

Impact of Prior Map Mismatch (Hypothesis 2): Figure 5 illustrates the effect of injecting noise into the agents' prior beliefs. Agents with access to the global potential fields with no lookahead (blue/green solid) are less negatively impacted than local (orange) agents with lookahead (dotted lines). It is worth noting in the prediction error graphs that global agents view the global state, but their local beliefs are not updated besides outside of comms usage.

Communication and Lookahead Utility (Hypothesis 3): Figure 6 shows that although $p = 0.05$ information sharing is enough to bound lookahead error regardless of the number of steps (1-10), naive SPF controllers fail to capitalize on the lookahead information, despite its accuracy.

6 DISCUSSION

While lookahead with zero communication increased MSE compared to stationary imagined teammates, we found the mean absolute error to be around 30% lower at lookahead 1 than 0. This

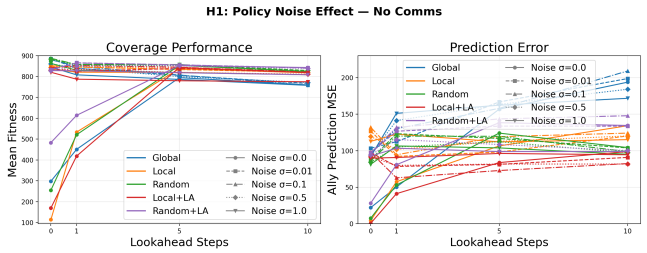


Figure 3: Policy Noise Weight No Comms: Color is the simulated annealing genome type, line-style is the noise level. Left is fitness (higher better) and right is ally location prediction error (lower better)

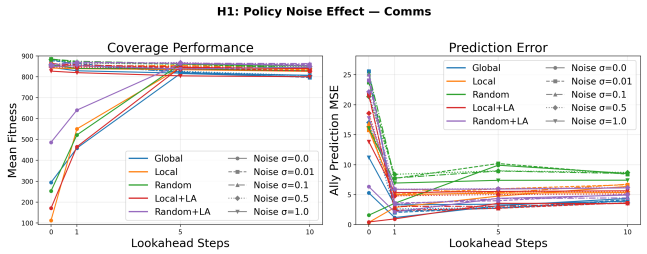


Figure 4: H1: Policy Noise Weight With Comms: The same scenario as Figure (3) but the agents have a 5 percent chance of sharing their up-to-date location with another agent at each frame. It can be seen that slight communication grounds all lookahead by limiting the random walk distance of compounding errors.

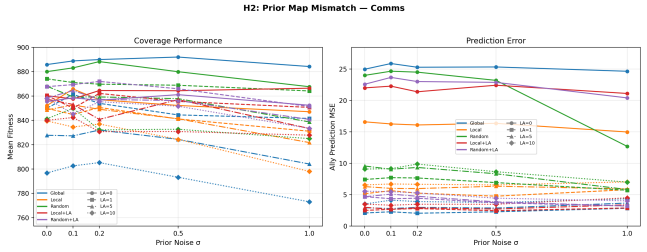


Figure 5: H2: Prior Map Noise Injection: The x axis is the level of injected noise, Color is simulated annealing genome and Line-Type is lookahead. Solid lines (less lookahead) are less impacted in terms of performance

property appears to originate from property ungrounded forward models suffer due to compounding spatial uncertainty ($\propto t^{3/2}$ for integrated random walks [10]) which rapidly exceeds the agent's actual possible traversal distance ($\propto t$)

For potential field navigation, noise can be necessary to push an agent out of areas with a zero gradient. We expected this noise to worsen prediction but that hypothesis was not supported. Interestingly, lookahead appears to act as an effective mechanism for perturbing the agent from a local minimum. We hypothesize that it is unlikely for all 4 agents in the belief state to be in local

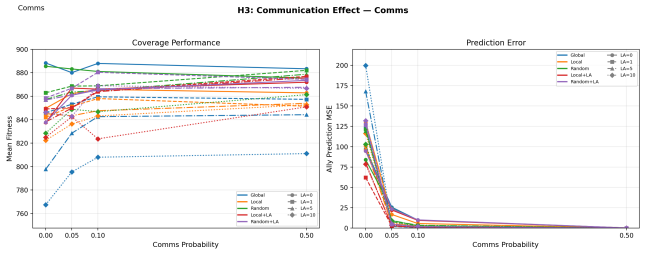


Figure 6: H3 Communication Probability: X-axis is communication probability, color is genome type, and line-style is lookahead. We see that Local (orange) and Local trained with lookahead (red) benefit the most from information sharing, but that 5 percent is sufficient.

minimums simultaneously, and that environment stochasticity as imagined agents "get stuck" with some probability are enough to perturb the potential field so that local minimums are transient.

We found that prior mismatch with communication probability above 0.05 had a negligible impact on performance. Across genomes, the distance drop-off exponent for danger was often 2 or 3, forcing the agent to only care about local danger because only local hazards can cause the agent harm. Because agent's act locally, far away updates are not very impactful beyond agent position updates. For communication probability, somewhere between $1/20$ and the time horizon of $1/1000$ leads to degradation of lookahead accuracy, but the frequency required to keep lookahead in check is relatively low.

It is important to note the scale limitations of our study: our 32×32 grid is quite course. Fine-grained prior noise or more space to wander may impact the magnitude's of our results.

7 CONCLUSIONS AND FUTURE WORK

This work demonstrates minimal communication ($p = 0.05$) combined with basic agent modeling bounds prediction error regardless of the number of projection steps for simple SPF controllers. While SPF shows robustness to both policy and environment noise, they fail to exploit the predictive information that lookahead provides, by focusing on local information. Potential field navigation is inherently reactive, so changing circumstances are not necessarily destructive to the control scheme.

Future work will involve additional search with teammate models designed for more pointed action selection, such as Monte Carlo Search, or as a lookahead/predictability baseline for deep reinforcement learning approaches. A field like "discovery value" to incentivize agents towards globally relevant objectives may also better highlight the effects of prior map mismatch. We hope that the introduction of a lightweight parallel environment with zero-copy memory sharing and realistic planetary terrain models will serve as a robust testbed for bridging classical heuristic navigation and advanced multi-agent reinforcement learning at a low computational cost. We put forth these initial results as a baseline from which higher-order navigation logic and models can be built and iterated on for a variety of coverage tasks.

REFERENCES

[1] Ross A Beyer, Oleg Alexandrov, and Scott McMichael. 2018. The Ames Stereo Pipeline: NASA's open source software for deriving and processing terrain data. *Earth and Space Science* 5, 9 (2018), 537–548.

[2] Ariyan Bighashdel, Daan De Geus, Pavol Jancura, and Gijs Dubbelman. 2024. Off-policy action anticipation in multi-agent reinforcement learning. *Journal of Machine Learning Research* 25, 67 (2024), 1–31.

[3] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[4] Feng Chen, Xinwei Chen, Rong-Jun Qin, Cong Guan, Lei Yuan, Zongzhang Zhang, and Yang Yu. 2025. Efficient Multi-Agent Cooperation Learning through Teammate Lookahead. *Transactions on Machine Learning Research* (2025).

[5] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. 2004. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation* 20, 2 (2004), 243–255.

[6] Jean-Pierre de la Croix, Federico Rossi, Roland Brockers, Dustin Aguilar, Keenan Albee, Elizabeth Boroson, Abhishek Cauligi, Jeff Delaune, Robert Hewitt, Dima Kogan, et al. 2024. Multi-agent autonomy for space exploration on the cadre lunar technology demonstration. In *2024 IEEE Aerospace Conference*. IEEE, 1–14.

[7] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant colony optimization. *IEEE computational intelligence magazine* 1, 4 (2006), 28–39.

[8] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[9] Roman Chiva Gil, Daniel Jarne Ornia, Khaled A Mustafa, and Javier Alonso Mora. 2024. Predictability Awareness for Efficient and Robust Multi-Agent Coordination. *arXiv preprint arXiv:2411.06223* (2024).

[10] Claude Godrèche and Jean-Marc Luck. 2022. Record statistics of integrated random walks and the random acceleration process. *Journal of Statistical Physics* 186, 1 (2022), 4.

[11] M. Hasan and R. Niyogi. 2025. Efficient Multi-Agent Exploration in Area Coverage Under Spatial and Resource Constraints. In *Proc. of ICAART*, Vol. 3. 1278–1287.

[12] O. Khatib. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research* (1986).

[13] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* 30 (2017).

[14] Alfred S McEwen, Eric M Eliason, James W Bergstrom, Nathan T Bridges, Candice J Hansen, W Alan Delamere, John A Grant, Virginia C Gulick, Kenneth E Herkenhoff, Laszlo Keszthelyi, et al. 2007. Mars reconnaissance orbiter's high resolution imaging science experiment (HiRISE). *Journal of Geophysical Research: Planets* 112, E5 (2007).

[15] Sharan Nayak, Grace Lim, Federico Rossi, Michael Otte, and Jean-Pierre de la Croix. 2025. Multi-robot exploration for the CADRE mission. *Autonomous Robots* 49, 2 (2025), 17.

[16] Kamal K Ndousse, Douglas Eck, Sergey Levine, and Natasha Jaques. 2021. Emergent social learning via multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 7991–8004.

[17] A. Pertzovsky, R. Stern, A. Felner, and R. Zivan. 2025. Enhancing Lifelong Multi-Agent Path-finding by Using Artificial Potential Fields. In *Proc. of AAMAS*.

[18] Kimon Protopapas and Anas Barakat. 2024. Policy mirror descent with lookahead. *Advances in Neural Information Processing Systems* 37 (2024), 26443–26481.

[19] Gregg Rabideau, Joseph Russino, Andrew Branch, Nihal Dhamani, Tiago Stegun Vaquero, Steve Chien, Jean-Pierre de la Croix, and Federico Rossi. 2025. Planning, scheduling, and execution on the Moon: the CADRE technology demonstration mission. *arXiv preprint arXiv:2502.14803* (2025).

[20] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[21] Mac Schwager, Daniela Rus, and Jean-Jacques Slotine. 2009. Decentralized, adaptive coverage control for networked robots. *The International Journal of Robotics Research* 28, 3 (2009), 357–375.

[22] Yuanbo Zhu, Guangjie Han, Chuan Lin, Fan Zhang, and Yun Hou. 2025. Multi-USV Coverage Path Planning Using Spatial Graph Multi-Actor-Attention-Critic Reinforcement Learning Framework With Operator Pooling. *IEEE Transactions on Mobile Computing* (2025).

Solving Organizational Multi-Robot Task Allocation Problems with Consensus-based Auctions

Victor Guillet
DTIS, ONERA, Université de Toulouse
Toulouse, France
victor.guillet@onera.fr

Christophe Grand
DTIS, ONERA, Université de Toulouse
Toulouse, France
christophe.grand@onera.fr

Charles Lesire
DTIS, ONERA, Université de Toulouse
Toulouse, France
charles.lesire@onera.fr

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

ABSTRACT

We propose a new approach to multi-robot task allocation that explicitly incorporates fleet organizational structures and constraints. We first formalize the Organizational Multi-Robot Task Allocation (Org-MRTA) problem, which extends classical MRTA by requiring allocations to respect organizational roles, missions, and norms in addition to performance. To address this problem, we introduce Org-CBBA, a hierarchical extension of the Consensus-Based Bundle Algorithm (CBBA), leveraging a MOISE+ organizational model to guide the allocation process to produce good quality solutions that remain organizationally coherent. This approach preserves the distributed robustness and convergence guarantees of CBBA while aligning task allocations with organizational doctrine. We evaluate Org-CBBA in synthetic mission scenarios inspired by real deployments and show that it maintains allocation quality while reducing computational overhead, while ensuring that solutions respect organizational requirements and constraints.

KEYWORDS

Multi-Robot Task Allocation; Organizational Modeling; Consensus-based Auctions

1 INTRODUCTION

Future space missions increasingly deploy fleets of autonomous and semi-autonomous assets, including satellite constellations and cooperative robotic teams. These missions demand rapid coordination and adaptability under severe communication and supervision constraints, rendering continuous centralized control impractical. Simultaneously, execution must comply with predefined command structures derived from planning, safety, and organizational doctrine. The organization is therefore intrinsic to mission design, and multi-agent systems must satisfy not only performance and robustness criteria but also explicit organizational constraints that delimit feasible allocations.

A central issue is the Multi-Robot Task Allocation (MRTA) problem, which assigns heterogeneous tasks to heterogeneous agents [4]. The objective is to optimize performance, such as minimizing mission duration or maximizing coverage, while accounting for capabilities, task dependencies, and environmental conditions. MRTA approaches range from centralized methods, which can provide globally optimal solutions but are unsuitable under unreliable communication, to distributed methods based on local decisions and peer-to-peer exchanges [11, 12]. The latter are generally more robust and operationally realistic.

Among distributed approaches, consensus-based algorithms are prominent. The Consensus-Based Bundle Algorithm (CBBA) [3] alternates local bundle construction with peer-to-peer consensus, ensuring distributed execution, robustness to communication failures, and convergence to a conflict-free allocation. However, CBBA and its variants are flat, with all agents bidding on all tasks, limiting scalability, and are role-agnostic, lacking explicit organizational structure.

A natural extension is hierarchical allocation. Existing solutions often cascade distinct algorithms [1], such as team-level followed by agent-level allocation, but this decoupling weakens integration. A unified hierarchical consensus mechanism can improve scalability by restricting bidding to role-relevant tasks and align allocations with explicit organizational models, enabling reasoning at multiple abstraction levels.

To represent the organization, we adopt MOISE+ [7], which specifies: (i) a structural model of roles, groups, and hierarchies; (ii) a functional model of missions, goals, and plans; and (iii) a deontic model linking roles to missions via permissions and obligations. This provides hierarchical abstraction and normative consistency.

We introduce Org-CBBA, a hierarchical consensus-based allocation algorithm embedding MOISE+ specifications into the allocation process. Org-CBBA preserves CBBA's distributed robustness and convergence guarantees while incorporating organizational abstraction and hierarchical workload distribution, producing scalable and organizationally coherent allocations. The paper makes three contributions: (i) the Org-MRTA model extending MRTA with organizational constraints; (ii) the Org-CBBA algorithm solving Org-MRTA; and (iii) an experimental evaluation on a scenario from a robotic challenge.

The remainder of the paper is structured as follows. Section 2 presents a motivating scenario; Section 3 reviews MRTA, consensus-based allocation, and MOISE+; Section 4 formalizes Org-MRTA; Section 5 details Org-CBBA; Section 6 reports experimental results in synthetic scenarios; and Section 7 concludes and outlines future work.

2 MOTIVATING MULTI-ROBOT SCENARIO

We consider a planetary surface exploration mission with a heterogeneous fleet of autonomous robots deployed from a central

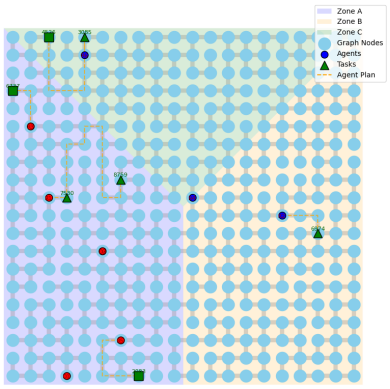


Figure 1: Environment and organizational layout.

landing site. Organizational roles and responsibilities are defined prior to deployment, while task allocation and execution occur autonomously onboard under intermittent and delayed communication with ground control.

At scenario onset, the fleet has stabilized the area around the base and operates in two geographically separated forward exploration zones extending in different directions. Transitions between zones are costly due to terrain and traversal constraints. Although no longer a primary exploration target, the base area remains operationally critical for shared logistics and support. Robots must therefore sustain exploration in both forward zones while collectively maintaining rear-area functions.

This setting induces a two-team organization: each team is primarily assigned to one forward zone to reduce costly cross-zone motion, while both share responsibility for logistics near the base. This structure captures mission planning trade-offs among spatial separation, coordination efficiency, and operational predictability under constrained communication.

From this abstract mission, we derive a simplified reference scenario that preserves the key spatial and organizational features. The objective is not to replicate a specific deployment, but to provide a controlled testbed for evaluating task allocation under explicit organizational constraints.

The environment (Figure 1) is represented as a partially connected 20×20 grid of nodes. A central landing site is located near the middle of the grid, from which all robots initially depart. The surface is divided into three exploration zones (A, B, and C), each associated with different operational assumptions: **Zone A (primary exploration zone):** the highest-priority region, characterized by the greatest density of exploration opportunities. Tasks in this zone primarily involve surface mapping, localized inspection, and anomaly characterization. **Zone B (secondary exploration zone):** a lower-intensity region requiring persistent monitoring and periodic exploration, including mapping and deployment of scientific instruments. **Zone C (shared exploration zone):** a tertiary exploration region with lower task density, representing partially explored areas that remain scientifically relevant. This zone has no fixed team ownership and serves as a shared operational space.

Organizational structure. To reflect mission-level planning, the fleet is divided into two organizational groups: *Surface Team A* and *Surface Team B*. The organizational hierarchy is strictly two-level: (1) *Surface Team A* is assigned responsibility for tasks arising in Zone A, while *Surface Team B* is assigned responsibility for tasks in Zone B; and (2) in Zone C, which represents a shared exploration region, tasks may be allocated to robots from either team. This structure enforces geographically scoped responsibilities while preserving flexibility in shared areas, providing a natural setting for evaluating how organizational constraints interact with decentralized consensus-based allocation.

Task model. Tasks appear dynamically and unpredictably throughout the mission, reflecting evolving scientific targets and environmental conditions. Their spatial distribution follows the zone priorities: the probability of task appearance is highest in Zone A, moderate in Zone B, and lowest in Zone C. Three task types are modeled, each requiring a specific capability:

- Mapping/Inspection tasks (O): reaching a location to acquire imagery or sensor data,
- Deployment tasks (T): reaching a location to deploy a scientific instrument or containment device,
- Anomaly-response tasks (I): reaching a location to inspect or approach a detected anomaly.

Although semantically distinct, all tasks are uniformly represented as go-to tasks: a robot must reach the designated node and possess the required skill. Tasks are modeled as independent, without sequential dependencies, allowing allocation decisions to be studied in isolation.

Fleet composition. The fleet is heterogeneous in skills but homogeneous in mobility. All robots move one step per time unit on the grid. The fleet is composed of: 3 robots capable only of mapping/inspection tasks (O), 2 robots capable of mapping and deployment tasks (O,T), 2 robots capable of mapping and anomaly-response tasks (O,I), 2 robots capable of deployment and anomaly-response tasks (T,I). Each robot is eligible only for tasks requiring skills it possesses, ensuring that allocation reflects heterogeneous capabilities.

During extended periods without ground contact, all task allocation decisions are performed onboard through inter-robot communication, subject to the organizational constraints defined above. This scenario therefore provides a representative testbed for studying hierarchical, decentralized task allocation for surface exploration missions under communication-limited conditions.

3 BACKGROUND AND RELATED WORK

This section introduces the Multi-Robot Task Allocation (MRTA) problem and its formal mathematical definition. We then review the main classes of approaches developed to solve MRTA, with a focus on decentralized consensus-based methods. Next, we present organizational modeling as a complementary paradigm for structuring multi-agent systems, with examples of different families. Finally, we discuss existing works that attempt to bridge MRTA and organizational models.

3.1 Multi-Robot Task Allocation Problem

The Multi-Robot Task Allocation (MRTA) problem [4] addresses the challenge of assigning a set of tasks to a team of robots in a

way that maximizes performance (e.g., mission success, reward) or minimizes cost (e.g., travel time, resource consumption). An allocation is valid if each task is assigned to at most one agent, and each agent is limited by its capacity (the maximum number of tasks it can handle).

Definition 1. *A* MRTA *problem* $\langle \mathcal{T}, \mathcal{A} \rangle$ *involves a set* $\mathcal{T}$ *of tasks and a set* $\mathcal{A}$ *of agents, and consists in assigning each task* $\tau_j \in \mathcal{T}$ *to an agent* $a_i \in \mathcal{A}$*, noted* $\tau_j \mapsto a_i$*, whilst maximizing some performance criteria.*

Considering each agent can be assigned at most L_τ tasks, and the maximum number of assignable tasks is $N_{\min} \triangleq \min\{|\mathcal{T}|, |\mathcal{A}| \cdot L_\tau\}$, MRTA can be expressed as follows [3]:

$$\max \quad \sum_{a_i \in \mathcal{A}} \left(\sum_{\tau_j \in \mathcal{T}} c_{ij}\left(\mathbf{x}_i, \mathbf{p}_i\right) x_{ij} \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{\tau_j \in \mathcal{T}} x_{ij} \leq L_\tau \quad \forall a_i \in \mathcal{A} \tag{2}$$

$$\sum_{a_i \in \mathcal{A}} x_{ij} \leq 1 \quad \forall \tau_j \in \mathcal{T} \tag{3}$$

$$\sum_{a_i \in \mathcal{A}} \sum_{\tau_j \in \mathcal{T}} x_{ij} = N_{\min} \tag{4}$$

$$x_{ij} \in \{0, 1\} \quad \forall a_i \in \mathcal{A}, \tau_j \in \mathcal{T} \tag{5}$$

Here, the decision variable $x_{ij} = 1$ if $\tau_j \mapsto a_i$, and 0 otherwise. The vector $\mathbf{p}_i \in (\{1, \ldots, |\mathcal{T}|\} \cup \{\emptyset\})^{L_t}$ represents agent i's ordered sequence of tasks (its path). The score function $c_{ij}(\mathbf{x}_i, \mathbf{p}_i) \geq 0$ encodes the reward for agent i to perform task j, and may depend on the agent's current task path. In mobile robotics, this function typically captures path-dependent metrics such as travel distance, energy cost, or mission completion time.

A wide variety of methods have been proposed for the MRTA problem, broadly falling into two categories. *Centralized approaches:* A single planner or auctioneer computes the allocation from global knowledge, often using combinatorial optimization (e.g., MILP, Hungarian algorithm [2]) or centralized auctions. While such methods yield high-quality or optimal solutions, they scale poorly and create a single point of failure, limiting their suitability for large-scale or communication-constrained settings. *Decentralized approaches:* Here, agents iteratively exchange bids or commitments to collectively converge to an allocation. These methods are more robust and scalable, as they avoid reliance on a central entity. Consensus-based algorithms form a key family, with the *Consensus-Based Bundle Algorithm* (CBBA) [3] being the most influential. CBBA alternates between local bundle construction and peer-to-peer consensus, and has been extensively studied for its scalability, robustness, and applicability in dynamic environments. Given their resilience to communication loss and scalability to larger teams, decentralised approaches are particularly relevant for multi-robot systems deployed in real-world, uncertain environments.

Additionally, many CBBA extensions exist for tackling different constraint types [1, 8, 9, 17]. Notably, intercession mechanisms (I-CBBA [5, 6]) provides a way for agents to bid on behalf of other agents, which we will leverage for our organizational extension.

3.2 Organizational Modeling and Allocation

As multi-robot systems scale, flat peer-to-peer coordination becomes inefficient and hard to interpret. *Organizational models* address this by structuring agent societies through roles, groups, authority, and norms, constraining autonomy in a purposeful way to ensure global objectives. The *Gaia methodology* [18] frames systems as organizations of roles, responsibilities, and protocols, offering both macro (societal) and micro (agent) views. However, Gaia assumes a *static* structure, making it more suited to design than runtime adaptation. Frameworks such as *OperA* [10, 16] distinguish between specification (roles, objectives, dependencies) and enactment (agents adopting roles). OperA supports *open systems*, where agents may join or leave dynamically, while tools like *OperettA* assist in design. Other approaches (e.g., *OMNI* [15]) combine structural and normative dimensions. In normative and institution-based models, norms (permissions, obligations, prohibitions) regulate agent behavior. Examples include *electronic institutions* and *HarmonIA* [14], which emphasize compliance and enforcement in heterogeneous or partially conflicting systems. Finally, *MOISE+* [7] integrates structural (roles, groups), functional (missions, goals), and deontic (permissions, obligations) layers. It supports complex hierarchical task structures and runtime reorganization, enabling flexibility while maintaining normative control, as needed in our scenario.

3.3 Focus on MOISE+

A MOISE+ *Organizational Specification* (OS) is defined as a tuple: $\text{OS} = \langle \text{SS}, \text{FS}, \text{DS} \rangle$, where SS is the *structural specification*, FS the *functional specification*, and DS the *deontic specification*. The *structural specification* $\text{SS} = \langle \mathcal{R}, \mathcal{SG}, \sqsubset, \mathcal{L}, C \rangle$ defines: a set of roles $\mathcal{R} = \{\rho_1, \ldots, \rho_{|\mathcal{R}|}\}$, a set of group specifications $\mathcal{SG}$, an inheritance relation $\sqsubset \subseteq \mathcal{R} \times \mathcal{R}$ ($\rho \sqsubset \rho'$ means ρ' specializes ρ), a set of links $\mathcal{L} = \{\ell_i(\rho_i^s, \rho_i^d, t), i \in [1..|\mathcal{L}|]\}$ with $t \in \{\text{acq}, \text{com}, \text{aut}\}$ denotes acquaintance, communication, or authority links, and a set of compatibility constraints C of the form $\rho_i \bowtie \rho_j$ indicating which roles may be jointly played. The *functional specification* $\text{FS} = \langle \mathcal{G}, \mathcal{M}, \mathcal{P}, mo, nm \rangle$ defines a set of goals $\mathcal{G}$, missions $\mathcal{M}$, and plans $\mathcal{P}$. Each mission $m \in \mathcal{M}$ maps to a set of goals via $mo : \mathcal{M} \to \mathbb{P}(\mathcal{G})$, and $nm : \mathcal{M} \to \mathbb{N} \times \mathbb{N}$ defines the cardinality of agents required to commit to each mission. The *deontic specification* defines the norms that link roles to missions, that is a set DS of permissions (resp. obligations) of the form $\text{per}(\rho, m, tc)$ (resp. $\text{obl}(\rho, m, tc)$), where $\text{per}(\rho, m, tc)$ grants permission for role ρ to commit to mission m under time constraint tc, and $\text{obl}(\rho, m, tc)$ imposes an obligation for role ρ to commit to mission m under time constraint tc. Inheritance ensures obligations and permissions are propagated along $\sqsubset$:

$$\text{obl}(\rho, m, tc) \Rightarrow \text{per}(\rho, m, tc) \tag{6}$$

$$\rho \sqsubset \rho' \wedge \text{obl}(\rho, m, tc) \Rightarrow \text{obl}(\rho', m, tc) \tag{7}$$

3.4 Bridging MRTA and Organizational Models

Although MRTA methods and organizational models emerged from distinct research traditions, they address complementary aspects of multi-robot coordination. MRTA focuses on algorithmic efficiency and conflict-free allocation of tasks, while organizational models emphasize structure, authority, and norms for scalable, interpretable cooperation. Several works have attempted to bridge these perspectives.

Early MRTA taxonomies recognized the role of coalitions and hierarchies in structuring allocation [4, 11]. Hierarchical team allocation and multi-stage pipelines have been studied [1], while

some works explicitly use organizational models to guide task allocation: for example, frameworks based on *OperA* incorporate role and organizational structure to determine who can/should perform which objectives [10]. *OperA*'s specification dimension (defining objectives, role dependencies) and enactment dimension (who plays what role) provide a basis for allocation constrained by organizational roles. Role-based methodologies (*Gaia* [18] and its extensions) provide design-time structure that can restrict possible agent-task mappings (via roles, permissions, protocols), which can be leveraged in MRTA to reduce search space, enforce constraints, or improve interpretability. Nevertheless, existing approaches often remain limited: they either rely on ad-hoc organizational scaffolding without a normative layer, or integrate task allocation only loosely into organizational formalisms. This motivates our work, which seeks to develop a tighter integration between consensus-based MRTA algorithms and formal organizational specifications, thereby bridging the gap between distributed task allocation and principled organizational modeling.

4 MODEL AND PROBLEM DEFINITION

Org-MRTA extends classical MRTA by embedding an explicit organizational model (based on MOISE+) that links roles, missions, and agents. This foundation allows us to formally define Org-MRTA as an extension of MRTA subject not only to performance criteria but also to organizational validity.

4.1 Instantiating MOISE+ for a Robot Fleet

The abstract organizational model from MOISE+ is instantiated onto a concrete fleet as follows. We first define a set of *agent classes* $\mathcal{AC}$, each associated with a subset of skills from the whole set of skills Σ via a mapping $\sigma : \mathcal{AC} \to \mathbb{P}(\Sigma)$. A set of *agents* $\mathcal{A} = \{a_1, \ldots, a_{N_a}\}$ is then introduced (same from the MRTA model), and we note $a_i \in AC_j$, with $AC_j \in \mathcal{AC}$, the fact that agent a_i is an instance of agent class AC_j. By extension to the mapping σ, we note $\sigma(a_i)$ the set of skills of agent a_i, inherited from its agent class. Each goal $g \in \mathcal{G}$ is annotated with a set of required skills $\kappa(g) \subseteq \Sigma$, and each task $\tau_j \in \mathcal{T}$ (same from the MRTA model) is an instance of a goal $g_k \in \mathcal{G}$, and thus inherits its skill requirements: $\kappa(\tau_j) = \kappa(g_k)$. Similarly, we define a set of *group instances* $\mathcal{G} = \{\gamma_1, \ldots, \gamma_{N_g}\}$, each being an instance of some group specification in $\mathcal{SG}$. An assignment relation $\pi : \mathcal{A} \times \mathcal{G} \to \mathbb{P}(\mathcal{R})$ specifies which agent plays which role in which group instance. A role ρ responsible for a set of goals $\mathcal{G}_\rho$ can be validly allocated to an agent $a \in \mathcal{A}$ though π only if $\bigcup_{g \in \mathcal{G}_\rho} \kappa(g) \subseteq \sigma(a)$.

This instantiation grounds the organizational specification in the fleet, linking abstract missions and goals to concrete agents, roles, and capabilities in a single model:

$$\text{OE} = \langle \text{OS}, \mathcal{T}, \mathcal{A}, \mathcal{AC}, \mathcal{G}, \Sigma, \sigma, \pi, \kappa \rangle \tag{8}$$

These specifications should ensure correctness at three levels: (i) that the organization structure and missions are well-formed with respect to MOISE+ constraints, (ii) that the fleet provides the skills required by the goals, and (iii) that assignments of agents to roles are feasible.

4.2 Organizational Multi-Robot Task Allocation

Definition 2. *An* Organizational Multi-Robot Task Allocation *(Org-MRTA) problem* $\langle OS, \mathcal{T}, \mathcal{A}, \mathcal{AC}, \mathcal{G}, \Sigma, \sigma, \pi, \kappa \rangle$ *is a classical MRTA problem* $\langle \mathcal{T}, \mathcal{A} \rangle$ *(Definition 1) provided with an instantiated organizational model OE (see Section 4.1). It consists in assigning each task* $\tau_j \in \mathcal{T}$ *to an agent* $a_i \in \mathcal{A}$*, noted* $\tau_j \mapsto a_i$*, while maximizing a set of performance criteria and satisfying the additional structural, functional, and deontic requirements imposed by the organization, such that:*

(1) a task τ_j *can be allocated to an agent* a_i *only if* $\kappa(\tau_j) \subseteq \sigma(a_i)$*,*
(2) a task τ_j *can be allocated to an agent* a_i *only if* $\exists \gamma \in \mathcal{G}, \exists \rho \in \pi(a_i, \gamma), \exists m \in \mathcal{M}, \exists tc$ *s.t.* $g \in mo(m)$ *and* $per(\rho, m, tc)$*,*
(3) a task τ_j *must be allocated to agent* a_i *if* $\exists \gamma \in \mathcal{G}, \exists \rho \in \pi(a_i, \gamma), \exists m \in \mathcal{M}, \exists tc$ *s.t.* $g \in mo(m)$ *and* $obl(\rho, m, tc)$*.*

Thus, Org-MRTA can be seen as a *constrained variant* of MRTA where feasible allocations are restricted by organizational roles, missions, and authority relations. These additional constraints ensure that the resulting allocation is not only optimal with respect to performance (e.g. travel distance, energy, completion time) but also *organizationally valid*.

Let us note that a solution to Org-MRTA is also a solution to the related MRTA, but the performances might differ due to the constraining organization (e.g. a complex organization might not be as efficient as a non-organized fleet for some criteria, such as the traveled distance, but better for other ones, such as the amount of exchanged messages). Moreover the allocation of roles specified in OE to agents is not an output of Org-MRTA. Org-MRTA focuses on task allocation, not role allocation. Finally, the current model does not exploit the planning dimension of the functional specification, since we again focus on task allocation, thus the missions and goals.

5 ORG-CBBA: ORGANIZATIONAL CONSENSUS-BASED BUNDLE ALGORITHM

Org-CBBA extends classical CBBA by embedding an explicit *organizational specification* into the allocation process. Instead of operating in a flat agent–task space, Org-CBBA exploits the hierarchical structure defined by MOISE+, linking roles, groups, missions, and goals. As a result, task allocation unfolds as a structured chain of decisions consistent with organizational constraints. This preserves the decentralized, consensus-driven nature of CBBA while ensuring allocations remain aligned with the organization.

5.1 Rationale

Formally, Org-CBBA embeds the instantiated organizational specification $\text{OE} = \langle \text{OS}, \mathcal{A}, \mathcal{AC}, \mathcal{G}, \Sigma, \sigma, \pi, \kappa \rangle$, where $\text{OS} = \langle \text{SS}, \text{FS}, \text{DS} \rangle$ (see Section 3.3, Section 4.1). At a high level, Org-CBBA operates through a *cascading sequence of auctions* that progressively narrow down responsibility for each task along the organizational hierarchy of groups of the fleet. The process unfolds as follows:

- At the *fleet level*, abstract coordination tasks are first auctioned among the fleet's top-level groups using a I-CBAA-style mechanism.
- The winning group becomes responsible for the task and, in doing so, generates a new abstract coordination task that

is auctioned at the next level of decomposition, where its child groups compete.
- The new abstract task for the substructure is then auctioned among its child groups through a I-CBAA-style auction.
- The process continues until it reaches the lowest-level group, and thus switches to a CBBA auction among the individual agents of that group. This final stage allocates the concrete execution of the task to a specific agent.

In this way, allocation answers the question — *which entity is responsible for this task?* — at progressively finer levels of resolution: from a top-level fleet group, to nested groups, and ultimately to individual agents. At the group level, bids are placed *on behalf of the group* by agents acting as its proxies through the *intercession mechanism* introduced in I-CBAA [5]. Group-level I-CBAA bids remain *lightweight and approximate*, but their resolution increases as auctions are performed over progressively smaller subsets of agents. I-CBAA is used at the group level since these auctions aim to assign responsibility rather than to sequence tasks. Task ordering is less relevant at this stage, as groups do not directly execute tasks—the temporal reasoning required for sequencing is deferred to the agent level, where CBBA auctions determine concrete allocations and execution plans.

As responsibility cascades downward, the bids therefore become more representative of the actual capabilities of the agents involved, culminating in the lowest-level CBBA auction where concrete task allocations are determined with full accuracy.

figure 2 illustrates the previously described process for the motivating multi-robot scenario, which is also later used in the experiment section.

5.2 Org-CBBA Specific Organizational Concepts

Org-CBBA requires defining a few additional notions and extending OE to incorporate key coordination elements necessary for its correct functioning.

In SS, a single abstract *proxy role* $\text{proxy} \in \mathcal{R}$ is introduced to enable group representation during inter-group interactions. Upon instantiation, this role can be played within any group $\gamma_k \in \mathcal{SG}$, thereby acting as the proxy for that specific group. Formally, if a_i acts as proxy in γ_k, i.e. $\text{proxy} \in \pi(a_i, \gamma_k)$. The proxy role is set to be compatible with all other roles, $\forall \rho \in \mathcal{R}, \text{proxy} \bowtie \rho$, ensuring that multiple agents can play it simultaneously.

A specific *allocation mission* $m^{\text{alloc}} \in \mathcal{M}$ dedicated to task delegation between hierarchical levels is defined in FS. This mission is composed of a new *allocation goal* $g^{\text{alloc}} \in \mathcal{G}$. We denote by $\mathcal{T}_{\text{alloc}}$ the set of allocation tasks. Each $\tau^{\text{alloc}_{jk}} \in \mathcal{T}_{\text{alloc}}$ is an instance of an allocation goal $g^{\text{alloc}} \in \mathcal{G}$. $\mathcal{T}$ is then the set of *concrete tasks*, corresponding to the executable tasks ultimately assigned to individual agents. Note that $\mathcal{T}_{\text{alloc}} \notin \mathcal{T}$ (thus $\mathcal{T}_{\text{alloc}}$ is effectively not part of the Org-MRTA problem).

We define the parents of a group γ as the set of all supergroups of γ following the inclusion relation: $\text{parents}(\gamma) = \{\gamma' \mid \gamma \subset \gamma' \text{ or } \exists \gamma'' \text{ such that } \gamma' \in \text{parents}(\gamma'') \text{ and } \gamma \subset \gamma''\}$. If $a \in \gamma$, then $\text{parents}(a) = \text{parents}(\gamma)$. Similarly, if $\tau^{\text{alloc}_{jk}}$ denotes the allocation task for task τ_j for group γ_k, and if $\gamma_k \in \text{parents}(\gamma_l)$, then $\tau^{\text{alloc}_{jk}}$ is the parent of $\tau^{\text{alloc}_{jl}}$ (τ_j if $a_l \in \mathcal{A}$). Consequently, for a given task $\tau^{\text{alloc}_{jk}} \in \mathcal{T}_{\text{alloc}}$, we define its set of parent tasks as: $\text{parents}(\tau^{\text{alloc}_{jk}}) = \{\tau^{\text{alloc}_{jl}} \mid \gamma_l \in \text{parents}(\gamma_k)\}$. For $\tau_j \in \mathcal{T}$, $\text{parents}(\tau_j) = \{\tau^{\text{alloc}_{jk}} \mid \gamma_k \in \mathcal{G}\}$. We then define $\text{subgroups}(\gamma)$ as the set of all subgroups included in γ.

If task $\tau_j \in \mathcal{T}$ requires allocation, an allocation instance $\tau^{\text{alloc}_{jk}}$ is created to determine the transfer of responsibility for τ_j from the parent group γ_k to one of its subgroups $\gamma_l \in \text{subgroups}(\gamma_k)$.

Finally, the proxy role is linked to m^{alloc} in DS through deontic relations of the form $\text{per}(\text{proxy}, m^{\text{alloc}}, tc)$ or $\text{obl}(\text{proxy}, m^{\text{alloc}}, tc)$, granting authority and responsibility to proxy agents to bid on behalf of their groups during distributed auctions.

5.3 Data Structures

Each agent maintains eight internal structures, identical to those defined in the I-CBBA framework [6] (as they can accommodate both CBBA and I-CBAA processes).

The *winning bid list* $\mathbf{y}_i$ (of size $|\mathcal{T}| + |\mathcal{T}_{\text{alloc}}|$) stores the highest bid value recorded across the fleet for each task, while the *winning agent list* $\mathbf{z}_i$ (also of size $|\mathcal{T}| + |\mathcal{T}_{\text{alloc}}|$) identifies the agent currently holding the winning bid for each task. The *timestamp list* $\mathbf{s}_i$ (of size $|\mathcal{A}| + |\mathcal{SG}|$) contains the most recent communication time from each agent, used to resolve consensus conflicts. Each agent also keeps its local *bundle* $\mathbf{b}_i$, representing the ordered list of tasks currently assigned to it, and the corresponding *path* $\mathbf{p}_i$, defining their execution order. In addition, the *fleet bids matrix* $\mathbf{f}_i \in \mathbb{R}^{(|\mathcal{T}|+|\mathcal{T}_{\text{alloc}}|)\times(|\mathcal{A}|+|\mathcal{SG}|)}$ stores, for each pair (j, r), the highest bid for task τ_j known by agent a_i to have been made by/for agent a_r. The associated *fleet priority matrix* ϕ_i records, for each bid in $\mathbf{f}_i$, the priority level of its emitter, as defined in the *priority vector* $|\mathbf{P}_\rho \in \mathbb{Z}^{|\mathcal{A}|+|\mathcal{SG}|}$, where each entry $\mathbf{P}_\rho(i)$ encodes the fixed priority level of agent a_i.

5.4 Org-CBBA Specificities

Each group $\gamma_k \in \mathcal{G}$ is represented in the bidding tables $(\mathbf{f}_i, \mathbf{y}_i, \mathbf{z}_i, \mathbf{s}_i, \phi_i)$ by a virtual entry that serves as the organizational placeholder for γ_k. Formally, if $\text{proxy} \in \pi(a_i, \gamma_k)$, then a_i intercedes on behalf of γ_k for $\tau^{\text{alloc}_{jk}}$ and is responsible to compute the corresponding bid value $\mathbf{f}_{ijk}$. Consequently, a task $\tau^{\text{alloc}_{jk}}$ can be assigned to a group γ, noted $\tau^{\text{alloc}_{jk}} \mapsto \gamma$.

Org-CBBA will exploit the chain of group inclusions, or *chain of affiliations*. Allocation unfolds top-down along these chains of affiliations. A concrete task $\tau_j \in \mathcal{T}$ may be allocated to an agent $a \in \mathcal{A}$ if there exists at least one path from a top-level allocation task $\tau^{\text{alloc}_{jk}}$ down to τ_j, which respect the following constraints:

(i) every parent group task in $\text{parents}(\tau_j)$ along the chain has been won by their respective group, and
(ii) the deontic rules $\text{per}(\rho, m, tc)$ or $\text{obl}(\rho, m, tc)$ hold for some mission m with $\tau_j \in mo(m)$.

In other words, if multiple affiliation paths exist, the agent is permitted to allocate τ_j to itself as long as *one* of these chains is satisfied. This ensures that task responsibility is consistently delegated from higher-level groups to lower-level groups and finally to individual agents, while allowing flexibility in cases where the organizational structure provides multiple valid paths. This mechanism is therefore not hard-coded for a particular organizational shape: it follows

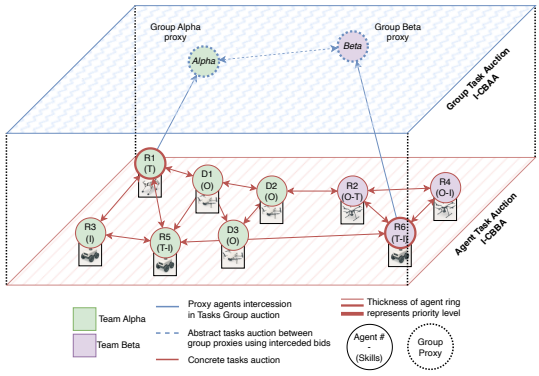


Figure 2: Illustration of Org-CBBA. At the group level, auctions for *allocation tasks* are voiced through proxy agents interceding on behalf of their groups. At the agent level, CBBA auctions allocate *concrete tasks* among individual agents.

the chains of affiliations specified in OE, thereby aligning the allocation process with the organizational structure provided, and it enables local checking of the validity of allocations, as follows:

Definition 3 (Organizational validity). *The allocation of a task τ to an agent a_i is* organizationally valid, *noted valid(τ, a_i) iff all these conditions hold:*

(i) τ is not already completed,
(ii) there exists ρ, m, tc, γ such that $\rho \in \pi(a_i, \gamma)$ and per(ρ, m, tc) or obl(ρ, m, tc) holds with $\tau \in mo(m)$, and
(iii) if $\tau \in \mathcal{T}$, $\kappa(\tau) \subseteq \sigma(a_i)$,
(iv) for each task $\tau^{alloc_{jk}} \in parents(\tau)$, $\tau^{alloc_{jk}} \mapsto \gamma \in parents(a_i)$

At each step of the allocation cascade, when responsibility is passed to a deeper level in the hierarchy, the winning bids are *refined* based on more detailed information from a smaller subset of agents. These refined values are then *backpropagated* upward along the affiliation chain to update the corresponding allocation task bids. If the refined information alters the outcome of any higher-level auction, the current winning branch must release the task and all its dependent tasks, and the allocation is redirected to the new winning group. This iterative correction ensures that higher-level approximations are continually aligned with progressively refined information, while preserving the fully distributed nature of CBBA.

Example 1 (Hierarchical allocation in Zone C). *Suppose a new observation task O_4 appears in Zone C. An allocation task A_C is created to decide whether Group Alpha or Group Beta (both $\in \mathcal{G}$) should take responsibility. Group representatives intercede on behalf of their groups to bid for A_C, using, e.g., the inverse Euclidean distance between the task location and their group centroid as a coarse proxy. If Group Alpha has the stronger centroid-based bid, it wins A_C. Only robots in Group Alpha then compete for O_4 with individual bids, subject to $\kappa(O_4) \subseteq \sigma(a)$ and permissions/obligations from DS.*

Example 2 (Refinement and backpropagation in Zone C). *Suppose Group Alpha initially wins the allocation task A_C for O_4. Within Group Alpha, all eligible robots bid for O_4 based on their distances. The winning robot's refined bid is backpropagated upward to update Alpha's effective bid for A_C. If this refined value is weaker than Group Beta's original bid, the allocation of A_C switches to Beta, forcing O_4 to be reallocated accordingly.*

5.5 Phase 1: Auction Process

The auction phase of Org-CBBA extends CBBA by embedding *organizational constraints* and handling both *allocation tasks* and *concrete tasks*. The process, summarized in Algorithm 1, unfolds as follows.

Step 1 [l7, l16] — Breadth-first parsing of tasks. Agents traverse the set of goals/tasks using a breadth-first strategy (`sortedBF`) along OE. Only valid allocations are considered (see Definition 3).

Step 2 [l8–l15] — Group coordination and bidding. This step is only active when both conditions hold: (i) $\tau_j \in \mathcal{T}_{\text{alloc}}$, i.e. τ_j is a $\tau^{\text{alloc}_{jk}}$, and (ii) a_i acts as a *group proxy* for group $\gamma_l \in \mathcal{G}$ for which $\gamma_l \in \text{subgroups}(\gamma_k)$, i.e., proxy $\in \pi(a_i, \gamma_l)$ and $\gamma_l \in \text{subgroups}(\gamma_k)$. In this case, a_i computes a bid $\mathbf{c}_{ijl}$ using BID^{gp} (the bidding logic used for group tasks) on behalf of its *group* γ_l through *bid intercession* [5, 6]. The resulting bid is integrated into the fleet bid matrix $\mathbf{f}$ using `merge` (l9). The `merge` *process* ensures that, for each conflicting bid emitted (if multiple agents hold the role proxy), the value and priority of the highest-priority ϕ_{ii} emitter are retained, systematically prioritizing higher-priority entries (see [6, Algorithm 2]).

a_i then updates $\mathbf{y}$ (winning bids), $\mathbf{z}$ (winning group), and $\mathbf{s}$ (timestamps) based on the outcome of the auction at that group level, *considering all bids currently present for τ_j*. Refined subgroup bids are then *backpropagated* upward (Algorithm 2) so that ancestor group tasks reflect the updated valuations of their subgroups. If this refinement changes a parent's outcome, the corresponding allocation is revised and, if necessary, re-auctioned. New group tasks may also be instantiated when previously unexplored ancestors are revealed in the organizational hierarchy.

Step 3 [l16] — Concrete task discovery. If τ_j is a concrete task, it is added to the available set $\mathcal{T}_i$ for bundle-building. Marginal gain computations are restricted to this set, reducing computation compared to evaluating all tasks.

Step 4 [l18–l27] — Bundle-building with marginal gains. On $\mathcal{T}_i$, bundle construction follows the standard CBBA procedure [3]. The bid for each task $\tau_j \in \mathcal{T}_i$ is determined using $\text{BID}^{\text{ind}}(j, i)$, which, as in the original CBBA algorithm, computes for every $j \in \mathcal{T}_i \setminus \mathbf{b}_i$ the marginal gain $\mathbf{c}_{ij}$ of inserting τ_j into $\mathbf{p}_i$ at the position yielding the largest increase in utility:

$$\text{BID}^{\text{ind}}(j, i) = \max_{n \le |\mathbf{p}_i|} \left(S_i^{\mathbf{p}_i \oplus_n \{j\}} - S_i^{\mathbf{p}_i} \right), \quad \forall j \in \mathcal{T}_i \setminus \mathbf{b}_i. \tag{9}$$

These values are merged into $\mathbf{f}$ with `merge`. Valid tasks $\mathbf{h}_i$ are determined by comparing with current winners, and bundle-building continues until no more tasks can be inserted.

5.6 Phase 2: Consensus Process

The consensus phase in Org-CBBA works almost exactly as in I-CBBA [6] (given we use the datastructures as defined in the I-CBBA algorithm): agents exchange and merge their local states $(\mathbf{y}_i, \mathbf{z}_i, \mathbf{s}_i, \mathbf{f}_i, \phi_i)$, update winning bids, and drop tasks if outbid. The only difference is that when a *group task* is dropped, any affiliated *concrete tasks* (and tasks following it in $\mathbf{b}$) already present in the agent's plan must also be dropped. This ensures that agents cannot

Algorithm 1: Org-CBBA Auction Phase for agent a_i

Procedure *BUILD BUNDLE*
$(\mathbf{y}_i(t-1), \mathbf{z}_i(t-1), \mathbf{b}_i(t-1), \mathbf{p}_i(t-1), \mathbf{f}_i(t-1), \phi_i(t-1))$
  $\mathbf{y}_i(t) = \mathbf{y}_i(t-1)$; $\mathbf{z}_i(t) = \mathbf{z}_i(t-1)$
  $\mathbf{b}_i(t) = \mathbf{b}_i(t-1)$; $\mathbf{p}_i(t) = \mathbf{p}_i(t-1)$
  $\mathbf{f}_i(t) = \mathbf{f}_i(t-1)$; $\phi_i(t) = \phi_i(t-1)$;
  $\mathscr{T}_i \leftarrow \emptyset$
  foreach $\tau_j \in$ sortedBF$(\mathscr{T} \cup \mathscr{T}_{alloc})$ *s.t. valid*(τ_j, a_i) **do**
    if τ_j *is a* $\tau^{alloc_{jk}}$ *and* proxy $\in \pi(a_i, \gamma_l)$ *and* $\gamma_l \in$ *subgroups*(γ_k) **then**
      $\mathbf{c}_{ijl} \leftarrow \text{BID}^{\text{gp}}(j, l)$
      merge$(\mathbf{f}_{ijl}(t), c_{ijl}, \phi_{ijl}(t), \mathbf{P}_\rho(i))$
      if $\max(\mathbf{f}_{ij}(t)) > \mathbf{y}_{ij}$ **then**
        $\mathbf{y}_{ij}(t) \leftarrow \max(\mathbf{f}_{ij}(t))$
        $\mathbf{z}_{ij}(t) \leftarrow$ winning representative
        $\mathbf{s}_{il}$ = current timestamp
        backpropagate$(\tau_j, \max(\mathbf{f}_{ij}(t)))$
        create_group_task(τ_j)
    else if τ_j *is a concrete task* **then** $\mathscr{T}_i \leftarrow \mathscr{T}_i \cup \{\tau_j\}$;
  while $|\mathscr{T}_i \setminus \mathbf{b}_i| > 0$ **do**
    $c_{iji} = \text{BID}^{\text{ind}}(j, i)$
    merge$(\mathbf{f}_{iji}(t), \mathbf{c}_{ij}, \phi_{iji}(t), \mathbf{P}_\rho(i)), \forall j \in \mathscr{T}_i \setminus \mathbf{b}_i$
    $\mathbf{h}_{ij} = \mathbb{I}\big(\mathbf{f}_{iji}(t) > \mathbf{y}_{ij}\big), \quad \forall j \in \mathscr{T}_i \setminus \mathbf{b}_i$
    if $|\mathbf{h}_i| = 0$ **then** terminate process // no valid tasks;
    $J_i = \text{argmax}_j\, h_{ij} \cdot \mathbf{f}_{iji}(t)$
    $n_{i,J_i} = \text{argmax}_n\, S_i^{\mathbf{p}_i \oplus_n \{J_i\}}$
    $\mathbf{b}_i \leftarrow \mathbf{b}_i \oplus_{\text{end}} \{J_i\}$
    $\mathbf{p}_i \leftarrow \mathbf{p}_i \oplus_{n_{iJ_i}} \{J_i\}$
    $\mathbf{y}_{iJ_i}(t) \leftarrow \mathbf{f}_{iJ_i i}(t)$
    $\mathbf{z}_{iJ_i}(t) \leftarrow i$
    backpropagate$(J_i, \mathbf{f}_{iJ_i i}(t))$

Algorithm 2: Backpropagation for task τ_j and value $\mathbf{c}$

Procedure backpropagate$(\tau_j, \mathbf{c})$
  foreach *parent* $\tau_k \in$ *parents*(τ_j) **do**
    update $\mathbf{f}_{ik}(t)$ with $\mathbf{c}$
    if *winning bid at* τ_k *changes* **then**
      update $\mathbf{y}_{ik}(t), \mathbf{z}_{ik}(t)$
      if $\tau_k \in \mathbf{b}_i$ **then**
        foreach $\tau_\ell \in \{$*tasks following* τ_k *in* $\mathbf{b}_i\}$ **do**
          $\mathbf{b}_i \leftarrow \mathbf{b}_i \setminus \{\tau_\ell\}$
          $\mathbf{p}_i \leftarrow \mathbf{p}_i \setminus \{\tau_\ell\}$
          $\mathbf{y}_{i\ell} \leftarrow 0$
          $\mathbf{z}_{i\ell} \leftarrow \varnothing$
    backpropagate(τ_k, c)

retain concrete tasks whose parent group task is no longer held, while leaving all other consensus mechanics unchanged.

5.7 Convergence and Optimality Guarantees

Org-CBBA inherits the convergence properties of its constituent mechanisms [3, section V-D]. At each branching point: (i) tasks are first allocated to groups via CBAA, which guarantees finite-time convergence to a unique winner, and (ii) within the winning group, CBBA is applied to allocate concrete tasks to individual agents, also guaranteeing convergence to a conflict-free allocation. Since these processes are chained hierarchically, the overall system converges to a consistent allocation in which each task is assigned to exactly one agent.

Bid backpropagation adjusts bid valuations without altering the consensus mechanics of CBAA/CBBA and thus does not affect convergence. For the mechanism to operate correctly, bids at group and agent levels must remain *comparable in magnitude*. If group-level bids are underestimated, backpropagation may fail to trigger needed reallocations; if agent-level bids are too small, every refinement would overturn higher-level results, collapsing the hierarchy into a flat CBBA. Hence, group-level bids should be *optimistic*—slightly overvaluing their agents' expected performance—so that backpropagation only corrects significant discrepancies. As in CBBA, Org-CBBA ensures convergence but not global optimality, producing locally optimal allocations consistent with OE.

6 EXPERIMENTAL EVALUATION

6.1 Driving scenario

Consider an initial setup where the 9 robots depart from the central base. Early in the mission, several tasks appear in Area A: for example, an observation task O_1, a trapping task T_1, and an interception task I_1. The robots eligible for each task bid according to their inverse Manhattan distance. For instance, O_1 can be claimed by any of the six robots possessing the O-skill, T_1 by any of the four robots with T-skill, and I_1 by any of the four robots with I-skill. The closest eligible robots win their respective bids, illustrating how spatial proximity and capability constraints jointly determine allocation outcomes in this motivating scenario.

6.2 Experimental Setup

To evaluate Org-CBBA, we compare two organizations, each representing a different way of structuring the decision process.

Flat organization (CBBA baseline). The fleet operates without hierarchy: every agent $a_i \in \mathscr{A}$ is considered for every task $\tau_j \in \mathscr{T}$ such that $\kappa(\tau_j) \subseteq \sigma(a_i)$. The bidding function computes the marginal cost of inserting a task into the current plan, using the *inverse of the shortest path length* as a distance-based utility, whilst meeting skill requirements. This corresponds to a direct application of the standard CBBA framework.

Hard hierarchy. The fleet is organized into two heterogeneous groups: Alpha (six robots: 3×O, 1×T, 1×I, 1×T,I) and Beta (three robots: 1×O,T, 1×O,I, 1×T,I). Group *Alpha* is exclusively responsible for tasks in Zone A, while Group *Beta* handles those in Zone B; neither group can handle tasks in the other's zone. This exclusivity is enforced through a logic-gate rule: if a task lies in Zone A, Alpha's bid is set to 1 and Beta's to 0, and symmetrically for Zone B. Both groups compete for each new concrete task τ_j appearing in the shared Zone C. Group-level bids for these allocation tasks are computed as the *inverse Euclidean distance* between the task location and the nearest agent of each group. The winning group then secures responsibility for the corresponding concrete task, which is subsequently allocated among agents at the robot level with a compatible skill set.

The mission environment is modeled as a 20×20 grid (400 nodes) with 70% of possible edges active while preserving graph connectivity. The home base is located at $(10, 10)$. Each simulation lasts 200 time steps, during which tasks appear dynamically.

The primary experimental variable is the *task spawning rate in Zone A*, which determines the relative workload distribution.

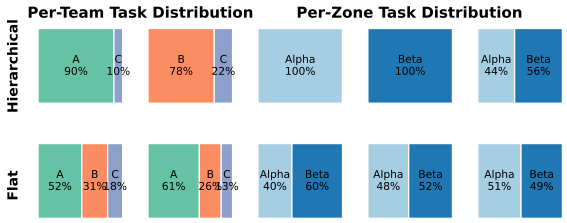


Figure 3: Task allocation distribution across organization type by groups and zones

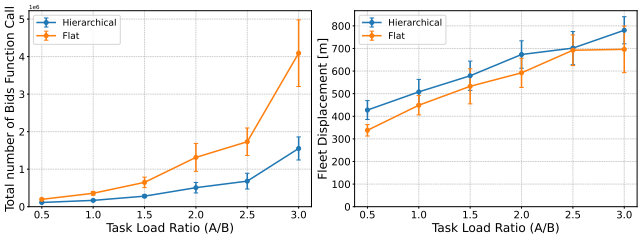


Figure 4: Bid count and total displacement vs ratio of task load in zone A and B

Table 1: Difference in cumulated displacement (across the fleet) and bid calls between hierarchical and flat configurations across task zone ratios.

Task Zone Ratio	Fleet Displacements (hierarchical − flat)	Bid Computations (hierarchical − flat)
0.5	89.50 (26.48%)	-82775.80 (-42.26%)
1	59.00 (13.14%)	-189356.60 (-52.95%)
1.5	46.60 (8.75%)	-367818.80 (-56.71%)
2	81.10 (13.69%)	-806959.50 (-61.42%)
2.5	9.30 (1.34%)	-1050260.60 (-60.66%)
3	84.50 (12.14%)	-2539189.60 (-62.07%)

Tasks in Zone B are generated at a constant rate across runs, while the rate in Zone A is progressively increased. Zone C maintains a constant task count. Varying the Zone A rate produces different task load ratios (A/B), enabling systematic comparison of organizational structures under distinct spatial workload distributions.

For each configuration, task arrival timelines are generated with randomized release times to capture deployment variability. Ten scenarios are simulated per case for statistical robustness.

Performance is evaluated along three dimensions: (i) *Cumulative fleet displacement*: total distance traveled by all agents, serving as an optimality metric. (ii) *Bid computations*: number of calls to each bidding logic, capturing computational overhead. (iii) *Per-zone dispatch*: fraction of tasks from Zones A, B, and C allocated to each group, measuring compliance with organizational responsibilities. We hypothesize that hierarchy reduces overhead by replacing numerous fine-grained bids with abstract group-level bids, progressively constraining the solution space while preserving allocation quality and organizational validity.

6.3 Results Analysis

The aggregated results across all task load ratios are shown in Figure 3. Values correspond to mean performance over ten randomized scenarios per configuration.

In Figure 3, the *Hierarchical* configuration enforces organizational constraints as designed: Team Alpha handles Zone A, Team Beta handles Zone B, and both share Zone C via group-level bidding. A slight asymmetry in Zone C favors Beta, explained by the higher task density in Zone A, which increases Alpha's workload and reduces its competitiveness in shared tasks. By contrast, the *Flat* configuration exhibits significant cross-zone mixing, as both teams compete across all zones without structural separation.

As reported in Table 1, the Hierarchical configuration incurs a moderate increase in cumulative fleet displacement relative to the Flat baseline (+1.3% to +26.5%), reflecting the limited optimality loss induced by zone exclusivity. However, bid evaluations decrease by 42–62%, confirming a substantial reduction in communication and decision-making load through group-level abstraction. As the task load in Zone A increases, computational savings grow while the displacement gap stabilizes below 15%, indicating that hierarchical decomposition becomes increasingly beneficial under spatially structured workloads.

Overall, introducing organizational structure ensures strict zone fidelity and markedly improves computational efficiency, with only limited degradation in allocation quality.

7 CONCLUSIONS

This paper introduced the Org-MRTA framework, which formalizes the trade-off between task allocation efficiency and organizational validity, ensuring that multi-robot systems operate within predefined structural and normative constraints. Such alignment is critical in domains where compliance with roles, missions, and authority relations is essential, including civil protection, military, and first-response operations.

To solve Org-MRTA in a distributed manner, we proposed Org-CBBA, which leverages the organizational structure to establish consensus between groups prior to consensus within the selected group. This hierarchical consensus mechanism was implemented and evaluated on a scenario derived from the `<anonymized>` challenge, designed to stress mission execution under unpredictable task arrivals. Compared to a flat CBBA without organizational structure, Org-CBBA achieves efficient allocations with an average optimality gap below 13%, while satisfying organizational constraints and significantly reducing bidding complexity, yielding more than 56% fewer bid computations.

Future work will focus on softening organizational constraints by allowing increased flexibility for robots to adapt under overload conditions, particularly when alternative groups are better suited to handle dense, unforeseen task arrivals. We also plan to evaluate Org-CBBA at larger scales, including the Robocup Rescue challenge [13], and to deploy it on a real robotic platform at ONERA for participation in the CoHoMa challenge.

ACKNOWLEDGMENTS

This work has been funded by the French Ministry of Defence's Innovation Agency (AID).

REFERENCES

[1] Matthew E. Argyle, David W. Casbeer, and Randal W. Beard. 2011. A Multi-Team Extension of the Consensus-Based Bundle Algorithm. *Proceedings of the 2011 American Control Conference* (2011), 5376–5381. https://api.semanticscholar.org/CorpusID:29143274

[2] Hamza Chakraa, François Guérin, Edouard Leclercq, and Dimitri Lefebvre. 2023. Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art. *Robotics and Autonomous Systems* 168 (2023), 104492. https://doi.org/10.1016/j.robot.2023.104492

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

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

[5] Victor Guillet, Christophe Grand, Charles Lesire, and Gauthier Picard. 2025. Bid Intercession to Unlock Human Control in Decentralized Consensus-Based Multi-robot Task Allocation Algorithms. In *Agents and Robots for reliable Engineered Autonomy*, Angelo Ferrando and Rafael C. Cardoso (Eds.). Springer Nature Switzerland, Cham, 99–114.

[6] Victor Guillet, Charles Lesire, Gauthier Picard, and Christophe Grand. 2025. Extending Consensus-based Task Allocation Algorithms with Bid Intercession to Foster Mixed-Initiative. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS-25)*. IFAAMAS, 932–940. https://www.ifaamas.org/Proceedings/aamas2025/pdfs/p932.pdf

[7] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. 2002. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In *Brazilian Symposium on Artificial Intelligence*. https://api.semanticscholar.org/CorpusID:7524187

[8] Simon Hunt, Qinggang Meng, Chris J. Hinde, and Tingwen Huang. 2014. A Consensus-Based Grouping Algorithm for Multi-agent Cooperative Task Allocation with Complex Requirements. *Cognitive Computation* 6 (2014), 338 – 350. https://api.semanticscholar.org/CorpusID:2626048

[9] Darren Hurley-Smith, Jodie Wetherall, Stephen R. Woodhead, and Andrew Adewunmi Adekunle. 2014. A Cluster-Based Approach to Consensus Based Distributed Task Allocation. *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* (2014), 428–431. https://api.semanticscholar.org/CorpusID:11026569

[10] Jie Jiang, Olivier Boissier, Virginia Dignum, and Javier Vázquez-Salceda. 2011. An Agent-Based Inter-Organizational Collaboration Framework based on OperA. In *In *Multiagent Systems and Applications / Workshops*, Lecture Notes in Artificial Intelligence*. Springer, 58–74.

[11] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. 2013. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Rob. Res.* 32, 12 (Oct. 2013), 1495–1512. https://doi.org/10.1177/0278364913496484

[12] Félix Quinton, Christophe Grand, and Charles Lesire. 2023. Market Approaches to the Multi-Robot Task Allocation Problem: a Survey. *Journal of Intelligent and Robotic Systems* 107, 2 (2023), 31. https://doi.org/10.1007/s10846-022-01803-0

[13] RoboCupRescue. 2025. RoboCupRescue Robot League. Retrieved October 6, 2025 from https://rrl.robocup.org/

[14] Javier Vázquez-Salceda and Virginia Dignum. 2003. HarmonIA: A New Approach to Model Electronic Institutions. In *The Role of Norms and Electronic Institutions in Multi-Agent Systems*, Javier Vázquez-Salceda (Ed.). IOS/Springer, 91–114.

[15] Javier Vázquez-Salceda, Virginia Dignum, and Olivier Boissier. 2003. The Role of Norms and Electronic Institutions in Multi-Agent Systems. In *Proceedings of the 2nd European Conference on Artificial Intelligence for Electronic Agents (EAAI or similar)*. IOS Press or Springer, 59–88. Also in "Electronic Institutions" / journal or book chapter.

[16] M. Weck and R. Dammertz. 1995. OPERA – A New Approach to Robot Programming. *CIRP Annals* 44, 1 (1995), 389–392. https://doi.org/10.1016/S0007-8506(07)62348-8

[17] Andrew K. Whitten, Han-Lim Choi, Luke B. Johnson, and Jonathan P. How. 2022. MIT Open Access Articles Decentralized Task Allocation with Coupled Constraints in Complex Missions. https://api.semanticscholar.org/CorpusID:8649938

[18] Michael Wooldridge, Franco Zambonelli, and Nicholas R. Jennings. 2000. The Gaia methodology for Agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems* 3, 3 (2000), 285–312. https://doi.org/10.1023/A:1010071910869

Decentralized Dynamic Task Allocation under Limited Communication Range

Alexandre Kha
Thales CortAIx-Labs, Lip6-CNRS
Palaiseau, France
alexandre.kha@thalesgroup.com

Aurélie Beynier
Lip6-CNRS
Paris, France
aurelie.beynier@lip6.fr

Christophe Labreuche
Thales CortAIx-Labs
Palaiseau, France
christophe.labreuche@thalesgroup.com

Mathieu Marchand
Thales CortAIx-Labs
Palaiseau, France
mathieu.marchand@thalesgroup.com

ABSTRACT

In Multi-Agent Systems (MAS), decentralized task allocation is an important topic that holds potential for scalability and robustness to failure points within the MAS. Classical decentralized approaches suppose that the communication graph of the agents is connected, namely that any two agents of the MAS are able to exchange information. However, this assumption is often invalid, especially when the agents have limited communication range and are moving. In this paper, we are interested in decentralized task allocation when agents have a limited communication range and receive dynamically tasks during the mission. We first introduce a procedure that extends decentralized algorithms to dynamic scenarios and limited communication range. However, agents that are not able to communicate between themselves may still plan on the same tasks and produce redundant executions, which is typically unfavorable to the allocation optimality. In response to this issue, we develop a novel algorithm leveraging generation of rendezvous points within the Consensus-Based Bundle Algorithm, that we call RDV-CBBA. This method enforces synchronization points to reduce conflicts between agents and thus task redundancy. Experiments show that RDV-CBBA outperforms significantly classical state-of-the-art methods extended to dynamic task allocation in terms of Total Distance (Min-Sum) objective, especially when the communication range is low.

KEYWORDS

Multi-Agent Systems, Decentralized Task Allocation, Dynamic Allocation, Auction-based Methods

ACM Reference Format:
Alexandre Kha, Aurélie Beynier, Christophe Labreuche, and Mathieu Marchand. 2026. Decentralized Dynamic Task Allocation under Limited Communication Range. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 9 pages.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

1 INTRODUCTION

Multi-Agent task allocation is pivotal for applications like Search and Rescue [30], disaster response [26], logistics [23], surveillance [13], and satellite scheduling [24], requiring efficient fleet coordination. A key challenge is developing decentralized approaches where agents compute allocations locally, providing scalability.

In this paper, we study a task allocation scenario where agents have limited communication range, i.e. they can only communicate with agents within their communication range, and are all periodically notified of the arrival of new tasks during the mission, and likely while they are traveling to complete previous tasks.

Prominent task allocation methods include Distributed Constraint Optimization Problems (DCOPs) [11, 26, 29], which optimize constraint functions that quantify the value of joint allocations among agent subgroups. Another class of methods consists of auction-based algorithms like the Consensus-Based Bundle Algorithm (CBBA) [7] and its extensions (ACBBA [16], PI [30], CBTA [28] etc.), which alternate between bidding and consensus phases. Alternatively, some approaches have extended the Hungarian Method [14, 27], efficiently matching agents to task clusters. Apart from DCOP algorithms, these methods generate disjoint (conflict-free) plans. However, they typically rely on the assumption of a connected communication graph, i.e. any two agents are able to communicate with each other, by eventually using other agents to relay their messages. They also assume that all tasks to service are known in advance by the agents.

In practice, mobility and limited communication ranges often fragment the network into disjoint "islands". This is critical in dynamic scenarios where scattered agents receive new tasks but lack the connectivity to negotiate conflict-free plans.

Previous work on limited communication often focuses on static tasks [2] or relies on opportunistic coordination [22]. However, the latter is driven by chance encounters and may be insufficient for dynamic scenarios with strict objectives. Synchronization strategies based on rendezvous have also been proposed. [8, 10]. Rendezvous methods gather agents closed enough such that they able to communicate and coordinate. Nevertheless, these approaches either require computational resources to determine a rendezvous location, or lack explicit optimality-driven rendezvous selection.

Our contributions are twofold: (1) We introduce a general dynamic allocation procedure, which is based on opportunistic encounters to allow agents to handle dynamic arrival of tasks and synchronize in a decentralized manner. (2) We propose RDV-CBBA (Rendezvous CBBA), a method that explicitly integrates the selection of rendezvous points into the decentralized allocation algorithm CBBA. RDV-CBBA enforces periodic connectivity by calculating optimal meeting points, using the Barycenter or the Fermat-Weber solution [3] for total distance. This ensures that agents connected at one instant regain connectivity later, allowing them to merge knowledge, resolve conflicts, and replan as tasks appear.

2 PROBLEM STATEMENT

Notations: We consider that usual operations on sets (e.g. $|.|, \cup, \cap, \setminus$) are valid on tuples, by assimilating them to the set of their elements.

The problem addressed in this paper is formulated as a dynamic Open Vehicle Routing Problem (OVRP), a variant of the VRP where agents are not required to return to the depot. This scenario closely models surveillance missions where a fleet of UAVs is initially deployed from a base. Subsequently, new points of interest are dynamically communicated by a mission center to all agents and must be visited while minimizing the cumulative travel time of the fleet (Min-Sum objective).

Task and Agents Definition

Let $\mathcal{I} = \{i_1, \dots, i_N\}$ be the set of agents in the problem. At each timestep t, we denote the positions of the agents as $\mathbf{x}_{i_1}(t), \dots, \mathbf{x}_{i_N}(t)$. Each agent i has a velocity v_i, meaning that between two timesteps t and $t+1$, any agent i can travel a distance of at most v_i.

Let $\mathring{\mathcal{J}}$ be the set of all possible tasks. At each timestep t, a finite set of new tasks $\mathcal{J}_{\text{app}}(t) = \{j_1(t), j_2(t), \dots, j_{k_t}(t)\} \subseteq \mathring{\mathcal{J}}$ appears, with cardinality $k_t \in \mathbb{N}$. By convention, $k_t = 0$ implies that $\mathcal{J}_{\text{app}}(t) = \emptyset$. We denote by $\mathcal{J}(t) = \bigcup_{t' \leq t} \mathcal{J}_{\text{app}}(t')$ the set of all tasks that have appeared up to timestep t. Furthermore, let t_j denote the appearance time of any task $j \in \mathring{\mathcal{J}}$. During the mission, agents complete tasks, and we denote the set of tasks completed by timestep t as $\mathcal{J}_{\text{done}}(t)$. Consequently, the set of actual tasks available at timestep t is given by $\overline{\mathcal{J}}(t) = \mathcal{J}(t) \setminus \mathcal{J}_{\text{done}}(t)$.

Agents are constrained by a communication range r. At any timestep t, an agent can only communicate directly with the other agents within its communication range. We define the Communication Graph $\mathcal{G}(t) = (\mathcal{V}(t), \mathcal{E}(t))$ where vertices correspond to the agents, i.e., $\mathcal{V}(t) = \mathcal{I}$. An edge exists between any two agents if they are within a distance of at most r, formally $\mathcal{E}(t) = \left\{(i,k) \in \mathcal{I}^2 \mid \|\mathbf{x}_i(t) - \mathbf{x}_k(t)\| \leq r\right\}$. We define *Connectivity Islands* (CI) as the connected components of this communication graph. Within each CI, there exists a path of edges linking any two agents, ensuring mutual reachability. We denote by $\mathcal{N}_i^+(t)$ the specific CI to which agent i belongs at timestep t, and by $D(t)$ the diameter of the graph $\mathcal{G}(t)$. Consequently, agents can only exchange information with others belonging to their current set $\mathcal{N}_i^+(t)$. While agents only communicate with neighbors within range r, messages are effectively propagated via multi-hop propagation to reach all agents in the island. An example of a MAS spatially partitioned into two CIs is presented in Figure 1.

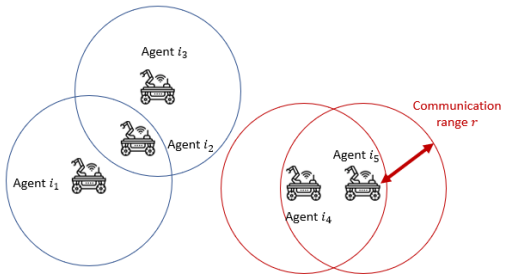


Figure 1: Two Connectivity Islands: $\{i_1, i_2, i_3\}$ and $\{i_4, i_5\}$. Agents i_1 and i_3 are not directly connected but belong to the same CI because Agent i_2 acts as a relay.

Optimization Problem

To model the optimization problem, we first introduce the following definitions. Let $\mathcal{T}_i(p_i)$ denote the time required for agent i to complete its path of tasks p_i.

Definition 1 (Admissible path). *For an agent $i \in \mathcal{I}$, a path is defined as a tuple $p_i = \left[j_1, \dots, j_{|p_i|}\right]$, where $j_\ell \in \mathring{\mathcal{J}}$ for all $\ell \in [\![1, |p_i|]\!]$. This path is admissible if and only if:*
(1) $j_\ell \neq j_m, \forall (\ell, m)$ s.t. $\ell \neq m$, and (2) $\mathcal{T}_i\left(\left[j_1, \dots, j_k\right]\right) \geq t_{j_k}, \forall k \in [\![1, |p_i|]\!]$. Constraint (1) ensures that an agent visits any specific task at most once. Constraint (2) ensures temporal causality, meaning an agent arrives at a task location only after the task has appeared. We denote the set of all admissible paths for any agent as $\mathcal{P}$.

Definition 2 (Admissible allocation). *An admissible path allocation $\boldsymbol{p} = \langle p_{i_1}, \dots, p_{i_N} \rangle$ is a tuple of joint admissible paths such that every task is covered, i.e., $\bigcup_{i \in \mathcal{I}} p_i = \mathring{\mathcal{J}}$. The set of all admissible allocations is denoted $\mathbb{P} \triangleq \mathcal{P}^{|\mathcal{I}|}$.*

Definition 3 (Conflict-free allocation). *An admissible allocation $\boldsymbol{p} = \langle p_{i_1}, \dots, p_{i_{N_a}} \rangle \in \mathbb{P}$ is conflict-free if $p_i \cap p_k = \emptyset$ for all distinct pairs $(i,k) \in \mathcal{I}^2$. Hence, each task is allocated to one agent.*

The task allocation optimization problem is defined as minimizing the total travel time (Min-Sum objective).

Problem 1 (Ideal Optimization Problem).

$$\min_{\boldsymbol{p} \in \mathbb{P}} \sum_{p_i \in \boldsymbol{p}} \mathcal{T}_i(p_i) \tag{1a}$$

$$s.t. \quad p_i \cap p_k = \emptyset, \qquad \forall (i,k) \in \mathcal{I}^2 \text{ s.t. } i \neq k. \tag{1b}$$

The objective function $\sum \mathcal{T}_i(p_i)$ represents the cumulative travel time of the fleet, which serves as a proxy for global energy consumption. Constraint (1b) *enforces a conflict-free allocation. Note that all tasks are covered as the space of solutions is $\mathbb{P}$ (cf. Definition 2).*

This problem formulation is "ideal" (or offline) because it assumes that agents possess a priori knowledge of all tasks $j \in \mathring{\mathcal{J}}$, including their appearance times, before the mission begins. In a realistic scenario, agents are notified of new tasks dynamically during the mission. Consequently, they must solve the allocation problem for these new tasks online, often while unable to communicate with other agents due to limited communication ranges. Therefore, in Section 4, we propose a decentralized method to solve online this dynamic problem efficiently.

3 RELATED WORK

3.1 Decentralized Task Allocation

Decentralized techniques for multiagent task allocation include Distributed Constrained Optimization Problems (DCOP) algorithms, which optimize utility sums among neighbors [5, 11, 29]. While suited for problems with high inter-dependencies, these methods are often myopic, preventing efficient path planning and are prone to conflicted allocation. Auction-based methods, such as the Consensus-Based Bundle Algorithm (CBBA) [7] typically alternate between a bidding phase and a consensus phase. CBBA guarantees convergence in a conflict-free allocation in finite number of iterations as well as 50% optimality for Diminishing Marginal Gain functions in fully-connected communication graphs. The Global CBBA (GCBBA) restores this guarantee for any communication graph and is also faster to converge [17]. Many more extensions have been designed to tackle more complex problems like communication asynchronicity (ACBBA [16]), coalitions (CBTA [28]), search and rescue (PI [30]), and human-guided allocation (I-CBBA [12]). Optimization-based algorithms like the Distributed-by-Matching Clones Hungarian-Based Algorithm (DMCHBA) [27] offer a very competitive alternative using first task clustering based on the Hungarian Method [20], and then routing. However, these algorithms primarily assume a connected communication graph at every timestep. In dynamic missions, as the network topology changes, conflict-freeness is lost and performance deteriorates because agents may do redundant actions.

3.2 Limited Communication and Dynamic Task Arrival

Cao et al. [4] analyzed the impact of packet loss on decentralized task allocation algorithms with dynamic task arrival. They found that CBBA maintains better optimality while Hungarian methods exhibit fewer conflicts. However, they did not address topological disruption caused by mobility, e.g. due to limited communication range. Ponda et al. [25] addressed time-constrained tasks and dynamic scenarios but relied on a mission center to select communicating subgroups, which does not enter our scope of study. More recently, Bai et al. [2] proposed the Distributed Auction Algorithm (DAA) for static batches, which merges plans when disjoint groups meet. Their approach relies on agents distinguishing between neighbors they had met previously and those they had not. However, relying on the "novelty" of neighbors is less relevant in dynamic settings because of the dynamicity of the set of tasks to execute.

3.3 Coordination via Rendezvous

The concept of rendezvous is well-established in control theory [6, 18, 21] and multi-agent exploration [9]. Relevant applications include: (1) Supply and Refueling: Do et al. [10] address a task allocation problem where supplier and receiver agents must rendezvous for battery recharging. Their method is centralized, first generating individual plans and subsequently selecting a rendezvous point from a candidate list to insert into the agents' trajectories. (2) Opportunistic Exploration: Luperto et al. [22] tackle area exploration with communication-constrained agents, and use backtracking to high-connectivity areas (e.g. corridors) to increase data exchange likelihood. (3) Synchronization Instants: Da Silva and Chaimowicz [8] treat exploration as a scheduling problem with random rendezvous points. While opportunistic encounters lack strict synchronization, explicit strategies like [10] ensure coordination but have not yet been adapted to dynamic auction mechanisms and may be computational depending on the size of the rendezvous candidates set. On another hand, Bai et al. [1] proposed a barycenter rendezvous for static scenarios and in a semi-centralized setting. Our work bridges this gap by calculating geometric rendezvous points (Barycenter, Fermat-Weber [3]) in a decentralized manner to forcefully synchronize agents for efficient replanning.

4 DECENTRALIZED DYNAMIC ALLOCATION FRAMEWORK

The ideal optimization problem formalized in Section 2 assumes that all tasks and their appearance times are known a priori. In this section, we present the practical online problem solved by the agents and introduce a general decentralized procedure to approximate the optimal solution of Problem 1 under dynamic conditions.

4.1 Practical Task Allocation Problem

At any timestep t, agents receive a new batch of tasks and must solve the following practical optimization problem:

Problem 2 (Practical Optimization Problem).

$$\max \sum_{i \in \mathcal{I}} \sum_{j \in \overline{\mathcal{J}}(t)} c_{ij}(p_i(t))\, x_{ij} \tag{2a}$$

$$\text{s. t.} \quad x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in \mathcal{I} \times \overline{\mathcal{J}}(t), \tag{2b}$$

$$\sum_{i \in \mathcal{I}} x_{ij} = 1, \qquad \forall j \in \overline{\mathcal{J}}(t), \tag{2c}$$

where the binary variable x_{ij} indicates whether task j is assigned to agent i, $p_i(t)$ is the current path of agent i, and $c_{ij}(p_i(t))$ represents the utility gain of adding task j to $p_i(t)$. To guarantee algorithmic convergence (see Section 5.2), c_{ij} serves as a proxy for the Min-Sum objective rather than its strict marginal cost. Equation (2c) *ensures all tasks are covered without conflicts.*

Crucially, agents cannot efficiently solve Problem 2 in a decentralized framework if the communication graph $\mathcal{G}(t)$ is not connected. Two disconnected Connectivity Islands may unknowingly allocate the same tasks, violating the conflict-free constraint, since they are unaware of each other's allocation. This leads to task redundancy and suboptimal performance. To address this, we present a procedure tackling the dynamic problem while mitigating inter-agent conflicts.

4.2 Dynamic Allocation Procedure

In the following analysis, we focus on the allocation procedure at timestep t and we omit the time dependency t for brevity. Recall that $\mathcal{N}_i^+$ denotes the CI to which agent $i \in \mathcal{I}$ belongs and let $\overline{\mathcal{J}_i} \subseteq \mathring{\mathcal{J}}$ denote the set of available tasks known to agent i. In this section, $\overline{\mathcal{J}_i} = \mathcal{J}_{\text{app}}$ for all $i \in \mathcal{I}$, i.e., it is a local copy of the appearing tasks.

We propose a novel dynamic allocation method (Algorithm 1) that extends the work of Bai et al. [2], initially for a single batch of tasks. The key features are:

(1) *Task Synchronization:* Upon connecting with a CI, agents communicate both their available and completed tasks to ensure coordination and prevent future redundancy (*Line 5*). They explicitly remove already completed tasks from their current paths.
(2) *Lazy Allocation:* Agents initially assign themselves the newly appearing tasks (*Line 4*). Reallocation is triggered using a conflict-free decentralized algorithm only when agents encounter a CI where task conflicts are detected (*Lines 7-9*). This strategy avoids unnecessary computations on already settled tasks, preserving resources and algorithm stability.

At the end of each timestep (*Line 10*), agents update their position $\mathbf{x}_i$ and their path p_i.

Algorithm 1 Dynamic allocation procedure for agent i

1: **procedure** Dynamic Agent i
2: **while** True **do** ▷ *Iterates for each timestep t*
3: $\overline{\mathcal{J}_i} \leftarrow \mathcal{J}_{\text{app}}$ ▷ *Receive new tasks from mission center*
4: $p_i \leftarrow$ Add tasks from $\overline{\mathcal{J}_i}$ to path
5: $\mathcal{N}_i^+, \bigcup_{k\in\mathcal{N}_i^+} p_k, \bigcap_{k\in\mathcal{N}_i^+} p_k \leftarrow$ Update the CI $\mathcal{N}_i^+$
6: ▷ *Check for conflicts within the CI*
7: **if** $|\bigcap_{k\in\mathcal{N}_i^+} p_k| > 0$ **then**
8: $p_i \leftarrow \emptyset$ ▷ *Reset p_i*
9: $p_i \leftarrow$ Run Allocation Algorithm within community $\mathcal{N}_i^+$ on tasks $\bigcup_{k\in\mathcal{N}_i^+} p_k$ ▷ *The algorithm is conflict-free*
10: $\mathbf{x}_i, p_i \leftarrow$ Move along path ▷ *Update position and path*

This procedure is generic and can wrap any decentralized task allocation algorithm (Line 9). The method relies on *opportunistic rendezvous* and does not strictly guarantee a conflict-free global state at all times. When new tasks appear while agents are disconnected, they may locally allocate the same tasks, leading to potential redundancy until connectivity is restored. Furthermore, this procedure requires agents to ping for neighbors at each timestep.

However, if agents are prohibited from re-taking a task they have previously conceded to another agent, convergence is guaranteed, as stated in the following theorem:

Theorem 1 (Convergence of Algorithm 1). *Given any conflict-free allocation method, and assuming that the total task set $\mathring{\mathcal{J}} = \bigcup_{t\geq 0} \mathcal{J}_{app}(t)$ is finite and agents are notified of the same task set $\mathcal{J}_{app}(t)$ at each timestep t, agents will eventually complete all tasks, possibly with intermediate conflicts.*

Proof. Let t be the time instant when all tasks in $\mathring{\mathcal{J}}$ have appeared. Since no further tasks arrive after t, the problem reduces to a static allocation on the remaining set $\overline{\mathcal{J}}(t)$. All agents initially plan routes for the remaining tasks $\overline{\mathcal{J}}(t)$ (cf. Line 4), ensuring coverage $\overline{\mathcal{J}}(t) \subseteq \bigcup_{i\in\mathcal{I}} p_i(t)$. For any agent i, the path p_i is modified only when meeting other agents. Crucially, any task released by agent i during conflict resolution is necessarily allocated to another agent (cf. Line 9). Hence, for any task $j \in \mathring{\mathcal{J}}$ and timestep $t' \geq t$:

$$j \in \left(\bigcup_{i\in\mathcal{I}} p_i(t')\right) \cup \mathcal{J}_{\text{done}}(t'). \tag{3}$$

The path $p_i(t)$ can only be modified a finite number of times, which is bounded by $|\mathcal{I}| - 1$. Indeed, if an agent meets any other agent in the MAS, applies a conflict-free allocation algorithm with them (Line 9), and without possibility to re-obtain a lost task, then it will reach conflict-free consensus with the whole MAS. Therefore, the path of each agent stabilizes after a finite number of modifications, allowing them to complete their assigned tasks. Let $t_{\text{end}} \geq t$ be the first timestep where all agents have completed their paths. From Equation (3), it follows: $\forall j \in \mathring{\mathcal{J}}, \quad j \in \mathcal{J}_{\text{done}}(t_{\text{end}})$. □

5 RENDEZ-VOUS CBBA (RDV-CBBA)

5.1 General Procedure

In Section 4, we have presented a general procedure extending any decentralized task allocation method to handle dynamic task arrival and limited communication range. However, this procedure fails to be conflict-free because agents from different CI may plan on the same tasks and not even meet to revise their plans. This impacts negatively performance since it generates unnecessary travels. In this section, we present the method RDV-CBBA to address this drawback. The RDV-CBBA framework exploits the fact that all agents belong to a single CI at the start of the mission. The core idea is to periodically restore this global connectivity to enable efficient coordination. We propose two variants of this approach: a *Myopic* version (Algorithm 3) and a *Proactive* version (Algorithm 4). In the Myopic variant, agents first meet and then plan a rendezvous point based on their allocated tasks. Conversely, in the Proactive variant, agents dynamically compute a future rendezvous point as soon as new tasks appear, before strictly needing to meet.

In this section, the available task set $\overline{\mathcal{J}_i}$ for any agent $i \in \mathcal{I}$ corresponds to the set of tasks arrived since previous rendezvous, i.e. for any timestep t, which previous rendezvous instant is $t_1 \leq t$, we have $\overline{\mathcal{J}_i}(t) = \bigcup_{t'\in[\![t,t_1]\!]} \mathcal{J}_{\text{app}}(t')$.

5.2 Principles of the Global CBBA (GCBBA)

We recall the principles of the GCBBA, an efficient method for static task allocation which is key to RDV-CBBA. The GCBBA is an improvement of the baseline CBBA by accelerating convergence and handling any type of connected communication graph in static task allocation [17]. In this subsection, let $\hat{\mathcal{J}}$ denote generally any batch of tasks to be allocated, and $\hat{\mathcal{I}}$ any CI. In this approach, any agent $i \in \hat{\mathcal{I}}$ iteratively places a bid $c_{ij}(p_i)$ on each task $j \in \hat{\mathcal{J}}$ (Auction phase) to then negotiate assignments with its peers (Consensus phase). Bids correspond exactly to the marginal utilities in Problem 2. The convergence of GCBBA is guaranteed within $|\hat{\mathcal{J}}|$ iterations, provided that the bidding function satisfies the *Diminishing Marginal Gain* (DMG) property, similar to submodularity in resource allocation [19].

Definition 4 (Diminishing Marginal Gain (DMG)). *A bidding function c_{ij} satisfies the DMG property if, for any agent $i \in \hat{\mathcal{I}}$, any path p_i, and any two distinct tasks $j, k \in \hat{\mathcal{J}} \setminus p_i$, the inequality $c_{ij}(p_i \oplus_\ell k) \leq c_{ij}(p_i)$ holds for all insertion indices $\ell \in [\![1, |p_i| + 1]\!]$, where $(p_i \oplus_\ell j)$ denotes the path p_i with task j inserted at position ℓ.*

The GCBBA procedure is outlined in Algorithm 2. The convergence detector used originally is omitted for simplicity, but convergence is still guaranteed. It utilizes the following variables and functions, largely inherited from standard CBBA:

Algorithm 2 Global Consensus-Based Bundle Algorithm (GCBBA)

procedure GCBBA
  $T \leftarrow 0$
  while $(T < |\hat{\mathcal{J}}|)$ **do**
    for $i \in \hat{\mathcal{I}}$ **do** ▷ ***Auction Phase***
      $(z_i, y_i, p_i) \leftarrow \texttt{ADD}(z_i, y_i, p_i)$
    for $t_{cons} \in [\![1, 2\,D]\!]$ **do** ▷ ***Consensus Phase***
      $T_{loc} \leftarrow 2 \cdot D \cdot T + t_{cons}$ ▷ *Global iteration marker*
      for $i \in \hat{\mathcal{I}}$ **do**
        $\texttt{SEND}(y_i, s_i)$ ▷ *Broadcast to neighbors* $\mathcal{N}_i$
        $\texttt{RECEIVE}((y_m)_{m\in\mathcal{N}_i}, (s_m)_{m\in\mathcal{N}_i})$
      for $i \in \hat{\mathcal{I}}$ **do** ▷ *Conflict Resolution*
        for $k \in \mathcal{N}_i$ **do**
          $(z_i, y_i, p_i) \leftarrow \texttt{RESOLVE}(y_i, y_k, s_i)$
          $s_i \leftarrow \texttt{S-UPDATE}(T_{loc}, s_i, s_k)$ ▷ *Update timestamps*
    $T \leftarrow T + 1$
  return $(p_i)_{i\in\hat{\mathcal{I}}}$

(1) $\mathcal{N}_i = \{k \in \hat{\mathcal{I}} \setminus \{i\} \mid \|\mathbf{x}_i - \mathbf{x}_k\| \le r\}$: The set of neighbors of agent i, defined by the communication range r.
(2) $y_i = (y_{ij})_{j\in\hat{\mathcal{J}}}$: A vector where $y_{ij} \in \mathbb{R}$ represents the highest bid for task j known to agent i.
(3) $z_i = (z_{ij})_{j\in\hat{\mathcal{J}}}$: A vector where $z_{ij} \in \hat{\mathcal{I}}$ indicates the winning agent for task j as perceived by agent i.
(4) $s_i = (s_{ik})_{k\in\hat{\mathcal{I}}}$: A timestamp vector where $s_{ik} \in \mathbb{N}$ records the last GCBBA iteration (variable T in Algorithm 2) in which agent i received information from agent k.
(5) ADD: A method that attempts to insert a task $j \in \hat{\mathcal{J}}$ into agent i's path at the optimal position, provided the marginal gain exceeds the current highest bid y_{ij}.
(6) RESOLVE: A conflict resolution method based on local communication. Whether to keep or release tasks is decided through a rule table (see [7, Table 1]).

The bidding function c_{ij} is critical for aligning the GCBBA solution with the Min-Sum objective (Problem 2). However, standard marginal costs derived directly from the Min-Sum objective $(\mathcal{T}_i(p_i \oplus_\ell j) - \mathcal{T}_i(p_i))$ do not satisfy the DMG condition (Definition 4), which is required for convergence.

While "warping" technique exists to artificially enforce DMG [15], it forces agents to misreport task valuations, which distorts the optimization landscape. Therefore, we adopt the *Total Path Bid* proposed in [17], which is proven to be DMG:

Definition 5 (Total Path Bid). *For any path p_i and task $j \in \hat{\mathcal{J}} \setminus p_i$, the Total Path bid is defined as:*

$$c_{ij}(p_i) = X - \min_{1\le\ell\le|p_i|+1} \mathcal{T}_i(p_i \oplus_\ell j), \tag{4a}$$

where X is a sufficiently large constant ensuring $c_{ij} > 0$.

Using this bidding scheme, the winning agent-task pair (i^*, j^*) selected at a GCBBA iteration satisfies the following equation: $(i^*, j^*) = \arg\min_{(i,j)\in\hat{\mathcal{I}}\times\hat{\mathcal{J}}} \mathcal{T}_i(p_i \oplus_{\ell^*} j)$. Consequently, this bid prioritizes agents capable of completing the task with the lowest total path duration. While this implicitly promotes load balancing, it serves as a robust proxy for the Min-Sum objective while maintaining the necessary theoretical convergence guarantees.

5.3 Myopic RDV-CBBA

First, we introduce Myopic RDV-CBBA, wherein agents meeting at a rendezvous (RDV) use GCBBA to allocate available tasks and determine a new RDV at the end of their paths. The Myopic RDV-CBBA procedure is detailed in Algorithm 3. Initially (or upon reaching a rendezvous), agents are co-located and fully connected (*Line 4*). They execute the GCBBA algorithm on the available tasks $\overline{\mathcal{J}_i}$ to generate conflict-free paths. Critically, they identify the set of "path-ending" tasks, denoted as $\mathcal{J}_{\text{last}} = \bigcup_{i\in\mathcal{I}} \{p_{i\,|p_i|}\}$ (*Line 5*). Based on these final task locations, the agents effectively compute a consensus rendezvous (RDV) point, such as the barycenter of $\mathcal{J}_{\text{last}}$, and append it at the end of their paths (*Lines 6-7*). Other methods for selecting this point are discussed in Section 5.6. Figure 2a describes an example of myopic rendezvous planning.

Agents then execute their paths (*Line 9*). The RDV is considered reached when agents are within a distance of $r/2$ from it, as it is sufficient to ensure full connectivity. Once synchronized at the RDV, the process repeats for any new tasks received during travel.

Algorithm 3 Myopic RDV-CBBA for agent i

procedure RDV-CBBA(M) AGENT i
  while True **do**
    $\overline{\mathcal{J}_i} \leftarrow \overline{\mathcal{J}_i} \cup \mathcal{J}_{\text{app}}$ ▷ *Receive new tasks from mission center*
    if (all $k \in \mathcal{I}$ at (RDV OR initial position)) **then**
      $p_i, \mathcal{J}_{\text{last}} \leftarrow$ GCBBA within $\mathcal{I}$ on tasks $\overline{\mathcal{J}_i}$
      RDV $\leftarrow$ Compute RDV based on $\mathcal{J}_{\text{last}}$
      $p_i \leftarrow p_i \oplus_{\text{end}}$ RDV ▷ *Append RDV to end of path*
      $\overline{\mathcal{J}_i} \leftarrow \emptyset$
    $\mathbf{x}_i, p_i \leftarrow$ Move along path

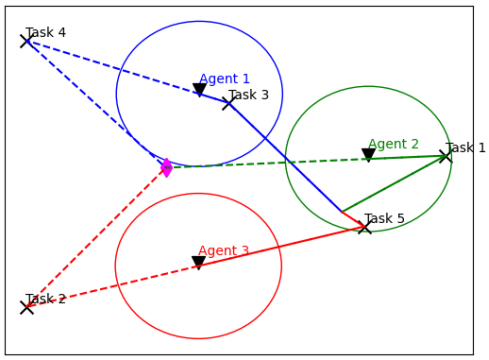
5.4 Proactive RDV-CBBA

The myopic rendezvous method allows the agents to agree on the next meeting point each time they meet. Before the agents arrive at this next meeting point, new tasks may have appeared, making this next meeting point inefficient if it is far from the new task locations. We hence propose a second version of RDV-CBBA, leveraging proactive rendezvous. In the Proactive procedure (Algorithm 4), agents initially generate a plan via GCBBA without immediately selecting a rendezvous location (*Lines 5-7*). While no new tasks appear, agents execute their paths normally. However, upon receiving a new batch of tasks, agents dynamically compute a RDV location based on the geometry of these *new* tasks and append it to their current plan (*Lines 9-10*).

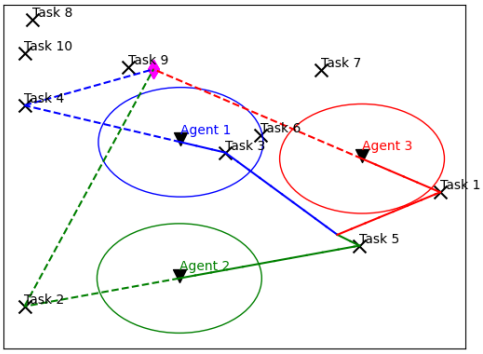
This approach allows agents to move towards the new task cluster before meeting, thereby anticipating future demand. This contrasts with the myopic strategy, which may pull agents back towards completed tasks. See an example of proactive rendezvous planning in Figure 2b. Once the RDV is reached, agents effectively re-plan, and the cycle continues.

5.5 Analysis

In this subsection, we provide the conditions for RDV-CBBA to converge under both myopic and proactive methods. Moreover, we discuss the success of convergence when new tasks are not simultaneously communicated to all agents.



(a) RDV-CBBA(M): RDV point (pink diamond) based on the barycenter of the *path-ending* tasks $\{1, 2, 4\}$.



(b) RDV-CBBA(P): RDV point (pink diamond) based on the barycenter of the *newly appearing* tasks $\{6, 7, 8, 9, 10\}$.

Figure 2: Comparison of Myopic and Proactive RDV-CBBA strategies. Triangles surrounded by a circle represent the agents and their communication range, crosses stand for tasks.

Algorithm 4 Proactive RDV-CBBA for agent i

1: **procedure** RDV-CBBA(P) AGENT i
2: **while** True **do**
3: $\overline{\mathcal{T}_i} \leftarrow \overline{\mathcal{T}_i} \cup \mathcal{J}_{\text{app}}$ ▷ *Receive new tasks from mission center*
4: **if** (all $k \in \mathcal{I}$ at (RDV OR initial position)) **then**
5: $p_i \leftarrow$ GCBBA within $\mathcal{I}$ on tasks $\overline{\mathcal{T}_i}$
6: $\overline{\mathcal{T}_i} \leftarrow \emptyset$
7: RDV $\leftarrow$ None
8: **else if** $(\overline{\mathcal{T}_i} \neq \emptyset)$ AND (RDV == None) **then**
9: RDV $\leftarrow$ Compute RDV based on $\overline{\mathcal{T}_i}$
10: $p_i \leftarrow p_i \oplus_{\text{end}}$ RDV
11: $\overline{\mathcal{T}_i} \leftarrow \emptyset$
12: $\mathbf{x}_i, p_i \leftarrow$ Move along path

THEOREM 2 (CONVERGENCE OF RDV-CBBA). *Let us assume that every agents are notified of the same task set $\mathcal{J}_{app}(t)$ at each timestep t. The Myopic RDV-CBBA allocates every task set sequence $(\mathcal{J}^t)_{t\geq 0}$ while guaranteeing a conflict-free solution. The Proactive RDV-CBBA guarantees the same properties, provided that agents independently compute the same rendezvous point.*

PROOF. For the *Myopic* version, the proof follows by induction. At $t = 0$, agents are connected and obtain a conflict-free allocation and a RDV location via GCBBA. Since the RDV is appended to the end of the path, agents are guaranteed to meet again. When the RDV is reached, the connectivity is restored, and the process repeats for any accumulated tasks.

For the *Proactive* version, the logic is similar but relies on the computation of a unique RDV point. It is necessary that all agents use a deterministic algorithm to identify rendezvous on their available tasks $\overline{\mathcal{T}_i}, \forall i \in \mathcal{I}$. If this holds, they meet at the intended location and proceed to the conflict-free allocation phase (Line 5). Else, they would wait indefinitely for neighbors at their respective RDV. □

It is important to note that in our study, at any timestep t, the mission center notifies **every agents** about the new tasks $\mathcal{J}_{\text{app}}(t)$. It implies that agents share at any t exactly the same available tasks $(\overline{\mathcal{T}_i})_{i\in\mathcal{I}}$, making these variables implicitly synchronized. If this does not hold, for example due to communication failure (lost message from the mission center, delayed messages ...), this impacts the two RDV procedures as follows. (1) Myopic RDV-CBBA: if at t, at least one agent receives the set $\mathcal{J}_{\text{app}}(t)$, then this set is shared among all agents at the rendezvous point (Line 5 of Algorithm 3), and planning is done as if all agents have received it. Theorem 2 is still valid for the myopic version. (2) Proactive RDV-CBBA: if at t, at least one agent does not receive $\mathcal{J}_{\text{app}}(t)$ when no RDV is yet selected, then agents will plan on at least two different RDV (Lines 9-10 of Algorithm 4). Theorem 2 fails for the proactive version. The **Myopic RDV-CBBA is hence theoretically much more robust** to communication failure.

5.6 Methods for Selecting Rendezvous

In this section, we present three distinct strategies for selecting rendezvous points. While the formulations below minimize travel distance, they can be trivially adapted to minimize travel time by incorporating agent velocities.

Depot Rendezvous: A trivial strategy is to designate the initial depot as the rendezvous point. While this approach requires no computation, it lacks adaptability to the spatial distribution of tasks and agents, likely inefficient if tasks appear far from the depot.

Barycenter Rendezvous: A geometric approach is to compute the barycenter (centroid) of a set of points. Let $\mathcal{S}$ be a set of positions of interest. In our application, $\mathcal{S}$ represents either the locations of the final tasks in the agents' current paths (Myopic) or the locations of a newly arrived task batch (Proactive). Identifying tasks with their spatial coordinates, the barycenter B is given by $B = \frac{1}{|\mathcal{S}|}\sum_{j\in\mathcal{S}} j$. It satisfies $B = \arg\min_{P\in\mathbb{R}^2} \sum_{j\in\mathcal{S}} ||j - P||^2$.

Fermat-Weber Rendezvous: The point that minimizes the sum of Euclidean distances to a set of points is the solution to the Fermat-Weber problem: $\min_{P\in\mathbb{R}^2} \sum_{j\in\mathcal{S}} w_j \cdot ||j - P||$, where w_j is a weight associated with point j. The solution is unique provided the points in $\mathcal{S}$ are not collinear. The Weiszfeld algorithm provides a robust approximation given an initial guess and a fixed number of iterations [3]. This solution is denoted FW. While the FW point theoretically

aligns with the Min-Sum objective, it does not guarantee global optimality for the combined allocation-routing problem. Indeed, simultaneously optimizing task allocation and rendezvous location is a much harder problem than solving them sequentially [10].

6 EXPERIMENTS

6.1 Description

The mission scenarios we consider take place in a rectangular domain defined by $[-20, 20] \times [-20, 20]$. Each agent $i \in \mathcal{I}$ has a uniform travel speed v_i DU/TU (Distance Unit / Time Unit), drawn from uniform distribution:

$$v_i \sim \mathcal{U}([5, 20]), \quad \forall i \in \mathcal{I}. \tag{5}$$

All agents initiate the mission from the depot at $(0, 0)$. We evaluate fleet sizes of $|\mathcal{I}| \in \{3, 4, 5\}$ and communication ranges of $r \in \{1, 3, 5\}$ DU. Between two timesteps t and $t + 1$, agents receive a new batch of tasks $\mathcal{J}_{\text{app}}(t)$ and may travel a maximum of v_i DU along their paths. In our study, 10 tasks are generated every 5 timesteps, for a total of 50 tasks. Hence, including the initial batch at $t = 0$, all tasks have arrived at $t = 20$.

Task positions are generated using two distinct processes. The first, denoted $\mathcal{D}_U$, uses a Uniform distribution (6). The second, denoted $\mathcal{D}_N$, uses a bivariate Normal distribution (7) to simulate an area of high task density.

$$j \sim \mathcal{U}_2([-20, 20]^2), \qquad \forall j \in \mathcal{J}, \tag{6}$$

$$j \sim \mathcal{N}_2\left(\begin{bmatrix} -10 \\ 10 \end{bmatrix}, \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}\right), \qquad \forall j \in \mathcal{J}. \tag{7}$$

Tasks generated by (7) are clipped to the mission area boundaries. The performance metrics evaluated are the Min-Sum objective (total travel time) and the number of conflicting task assignments, averaged over 100 random scenarios.

6.2 Comparing Rendezvous Methods

We refer to the RDV-CBBA variants using Depot, Barycenter, and Fermat-Weber rendezvous points as Depot, Bary, and FW, respectively. We distinguish the Myopic and Proactive versions with the suffixes (M) and (P) (e.g., FW(M) and FW(P)). For these experiments, the communication range is fixed at $r = 1$ DU. The FW rendezvous location is calculated using the Weiszfeld Algorithm [3] initialized at the barycenter, with 1000 iterations. For the Myopic version (FW(M)), the weight associated with each path-ending task $j_i^{\text{last}} \in \mathcal{J}_{\text{last}}$ is $w_i = 1/v_i$ (prioritizing slower agents). For the Proactive version (FW(P)), all weights are equal to 1. Table 1 presents the average Min-Sum values for both task distributions.

Algo	Fleet size	Min-Sum ($\mathcal{D}_U$)	Min-Sum ($\mathcal{D}_N$)
Depot(M)	3 / 5 / 10	61.0 / 69.7 / 68.3	55.8 / 58.7 / 59.2
Bary(M)	3 / 5 / 10	58.4 / 69.6 / 74.8	46.7 / 54.4 / 59.4
FW(M)	3 / 5 / 10	**57.1 / 69.3 / 69.8**	**46.1 / 53.1 / 57.6**
Bary(P)	3 / 5 / 10	**55.8** / **65.0** / **73.5**	**45.2** / **50.6** / **57.6**
FW(P)	3 / 5 / 10	56.2 / 65.3 / 74.6	45.2 / 50.8 / 60.3

Table 1: Min-Sum comparison of RDV strategies.

The results indicate comparable performance across all rendezvous methods when tasks are uniformly distributed, with FW(M) and Bary(P) performing best by a narrow margin. The Depot rendezvous also performs reasonably well in this case because the center of the area coincides with the expected value of the uniform distribution.

For tasks drawn from $\mathcal{D}_N$, the Depot rendezvous is ill-suited, as it forces agents to return to the center of the arena rather than remaining in the zone of high task density. Conversely, FW(M) and Bary(P) adapt better, placing the rendezvous within this zone.

FW(M) maintains a competitive performance with Bary(P) as well as superior robustness in practical communication settings (see Section 5.6). We reference it simply as RDV-CBBA and use it as the baseline for the subsequent comparative experiments.

6.3 Comparison with Dynamic Task Allocation Baselines

In this section, we compare the performance of RDV-CBBA against three state-of-the-art decentralized methods adapted for dynamic scenarios using Algorithm 1: GCBBA [17], the PI heuristic [30], and DMCHBA [27]. PI is an extension of CBBA designed for complex objective functions and exploration, while DMCHBA utilizes a distributed Hungarian method followed by a TSP routing phase. For GCBBA and PI, we use the bidding scheme defined in Equation (4) and a greedy insertion heuristic for path creation.

Figures 3 and 4 present the Min-Sum values and the number of task conflicts, respectively, for tasks generated via $\mathcal{D}_U$. We also include a lower bound labeled **UR-GCBBA** (Unlimited Range GCBBA), representing the performance of GCBBA in a fully connected network where conflict-free allocation is guaranteed.

Min-Sum Performance: When the communication range is strictly limited ($r = 1$), the dynamic algorithms GCBBA, PI and DMCHBA perform poorly, with performance deteriorating as the fleet size increases due to high task redundancy (multiple agents completing the same task). As the communication range increases, their performance improves, approaching that of RDV-CBBA. However, for $|\mathcal{I}| = 10$ and $r = 5$, they remain approximately 85% worse than RDV-CBBA. Although DMCHBA performs well in static, connected scenarios [27], it is severely impacted by communication constraints in the dynamic setting. This performance drop aligns with findings by Cao et al. [4] for a similar Hungarian-based approache under non-ideal communication. GCBBA and PI exhibit comparable performance, likely due to the use of the same DMG bidding scheme. RDV-CBBA demonstrates stable performance that is largely independent of r, as it relies on physical rendezvous rather than continuous communication. However, UR-CBBA still outperforms RDV-CBBA by at least 50% for 10 agents, indicating that while RDV-CBBA effectively mitigates conflicts, there is still a traveling cost for achieving synchronization at the RDV compared to perfect instantaneous communication.

Conflict Analysis: The number of conflicts for the dynamic GCBBA, PI and DMCHBA decreases as r increases but remains significant even at $r = 5$ (approximately 70 conflicts for 10 agents). DMCHBA produces fewer conflicts than GCBBA and PI, a trend also noted in [4]. Crucially, as guaranteed by Theorem 2, RDV-CBBA provides

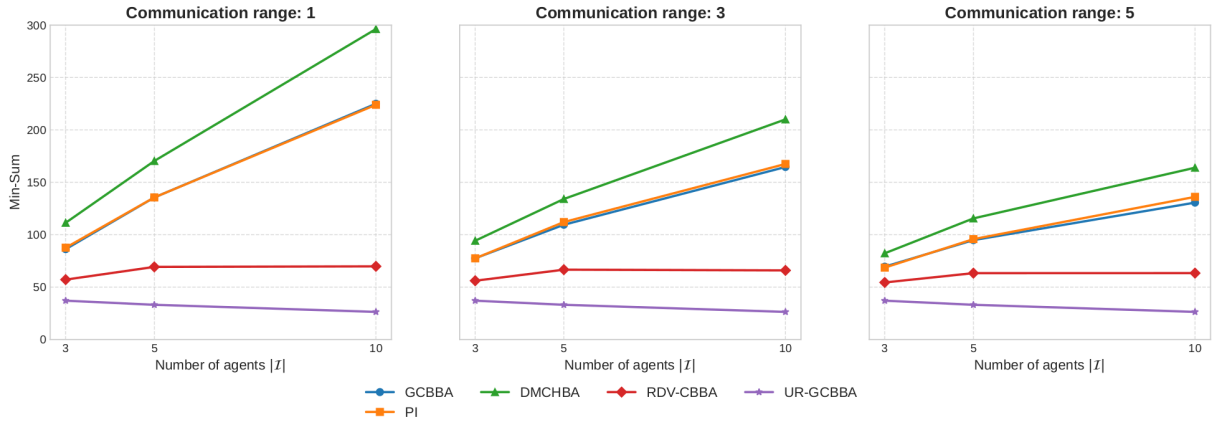


Figure 3: Min-Sum comparison of dynamic CBBA, PI, DMCHBA and RDV-CBBA.

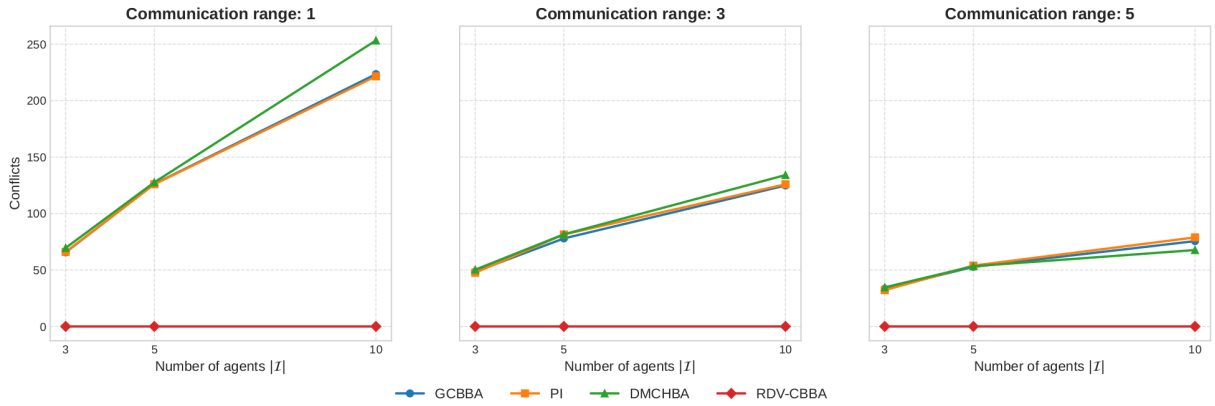


Figure 4: Number of conflicting task assignments for dynamic CBBA, PI, DMCHBA and RDV-CBBA.

zero-conflict allocations whereas DMCHBA may return allocation with conflict. RDV-CBBA thus prevents the agents from executing the same tasks several times unnecessarily.

7 CONCLUSION

In this paper, we addressed the problem of decentralized task allocation in dynamic environments where agents are constrained by limited communication ranges. This setting violates the fundamental assumption of network connectivity required for the convergence of classical decentralized algorithms.

First, we proposed a general framework to extend classical decentralized algorithms to dynamic scenarios. This method employs a "lazy" allocation strategy: agents bid only on newly appearing tasks and trigger reallocation only when conflicts are explicitly detected with neighbors. While this approach ensures finite-time completion of tasks, it is inherently *conflict-prone*. In low-communication settings, the lack of coordination opportunities leads to high task redundancy and suboptimal performance.

To overcome this limitation, we introduced **RDV-CBBA**, a novel approach that enforces periodic synchronization at computed rendezvous points to guarantee conflict-free allocations. We developed two variants: a *Myopic* strategy, where the rendezvous is derived from the current path endpoints, and a *Proactive* strategy, which anticipates future RDV based on new task clusters. Our analysis reveals a critical trade-off: while the Proactive strategy offers marginal performance gains, it is brittle in the presence of inconsistent information (e.g., packet loss). The Myopic strategy, conversely, is robust as it relies on consensus formed during physical meetings, making it the preferable choice for realistic deployments.

Experiments indicate that RDV-CBBA significantly outperforms state-of-the-art dynamic extensions of CBBA, PI, and DMCHBA in terms of the Min-Sum objective and conflict avoidance across all tested communication ranges. However, a performance gap remains compared to the theoretical lower bound of unlimited communication (UR-GCBBA), indicating potential for further optimization.

In future work, we aim to evaluate RDV-CBBA in practical simulations with non-ideal communication layers (lost messages, delays). Moreover, our RDV methods assume that all agents eventually meet at an intended RDV. Additional experiments could be to introduce agent failures. Furthermore, we plan to investigate coupled optimization approaches that solve the task allocation and rendezvous location problems simultaneously, rather than sequentially, to further close the optimality gap.

REFERENCES

[1] Xiaoshan Bai, Weisheng Yan, Ming Cao, and Jie Huang. 2017. Target assignment for robots constrained by limited communication range. *arXiv preprint arXiv:1702.04700* (2017).

[2] Xiaoshan Bai, Weisheng Yan, and Shuzhi Sam Ge. 2021. Distributed task assignment for multiple robots under limited communication range. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52, 7 (2021), 4259–4271.

[3] Amir Beck and Shoham Sabach. 2015. Weiszfeld's Method: Old and New Results. *Journal of Optimization Theory and Applications* 164, 1 (2015), 1–40. https://ideas.repec.org//a/spr/joptap/v164y2015i1d10.1007_s10957-014-0586-7.html Publisher: Springer.

[4] Yan Cao, Teng Long, Jingliang Sun, Zhu Wang, and Guangtong Xu. 2023. Comparison of Distributed Task Allocation Algorithms Considering Non-ideal Communication Factors for Multi-UAV Collaborative Visit Missions. *IEEE Robotics and Automation Letters* (2023), 1–8. https://doi.org/10.1109/LRA.2023.3295999 Conference Name: IEEE Robotics and Automation Letters.

[5] Jesús Cerquides, Rémi Emonet, Gauthier Picard, and Juan A Rodríguez-Aguilar. 2018. DECIMAXSUM: Using decimation to improve max-sum on cyclic DCOPs. In *Artificial Intelligence Research and Development*. IOS Press, 27–36.

[6] Zhekun Cheng, Liangyu Zhao, and Zhongjiao Shi. 2022. Decentralized Multi-UAV Path Planning Based on Two-Layer Coordinative Framework for Formation Rendezvous. *IEEE Access* 10 (01 2022), 45695–45708. https://doi.org/10.1109/ACCESS.2022.3170583

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

[8] Alysson Ribeiro da Silva and Luiz Chaimowicz. 2025. Intermittent Rendezvous Plans with Mixed Integer Linear Program for Large-Scale Multi-Robot Exploration. arXiv:2511.12237 [cs.RO] https://arxiv.org/abs/2511.12237

[9] Julian de Hoog, Stephen Cameron, and Arnoud Visser. 2010. Selection of Rendezvous Points for Multi-Robot Exploration in Dynamic Environments.

[10] Haggi Do, Junwoo Jang, and Jinwhan Kim. 2025. Heterogeneous multi-robot system mission planning with cooperative replenishment through data-driven rendezvous point selection. *Intelligent Service Robotics* 18, 1 (2025), 61–73.

[11] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed Constraint Optimization Problems and Applications: A Survey. *Journal of Artificial Intelligence Research* 61 (March 2018), 623–698. https://doi.org/10.1613/jair.5565

[12] Victor Guillet, Christophe Grand, Charles Lesire, and Gauthier Picard. 2025. Extending Consensus-based Task Allocation Algorithms with Bid Intercession to Foster Mixed-Initiative. In *24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-25)*. International Foundation for Autonomous Agents and Multiagent Systems, Detroit (Michigan), United States, 932–940. https://doi.org/10.5555/3709347.3743612

[13] Sara Hsaini, Rabah Ammour, Leonardo Brenner, Moulay El Hassan Charaf, and Isabel Demongodin. 2023. A Decentralized Based Approach Using Hybrid Filtered Beam Search Algorithm for Monitoring Patrols. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 1436–1441.

[14] Sarah Ismail and Liang Sun. 2017. Decentralized hungarian-based approach for fast and scalable task allocation. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 23–28. https://doi.org/10.1109/ICUAS.2017.7991447

[15] Luke Johnson, Han-Lim Choi, Sameera Ponda, and Jonathan P How. 2012. Allowing non-submodular score functions in distributed task allocation. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 4702–4708.

[16] Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan How. 2011. Asynchronous Decentralized Task Allocation for Dynamic Environments. In *Infotech@Aerospace 2011*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2011-1441 _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2011-1441.

[17] Alexandre Kha, Aurélie Beynier, Christophe Labreuche, and Mathieu Marchand. 2025. Enhancing CBBA convergence and optimality guarantees in Multiagent Task Allocation. In *hal-05218405*. https://hal.science/hal-05218405

[18] Jaekwang Kim, Hyung-Jun Park, Aditya Penumarti, and Jaejeong Shin. 2024. Fast Marching Based Rendezvous Path Planning for a Team of Heterogeneous Vehicles. *IEEE Access* PP (01 2024), 1–1. https://doi.org/10.1109/ACCESS.2024.3444314

[19] Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. *Tractability* 3, 71-104 (2014), 3.

[20] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[21] J. Lin, A. S. Morse, and B. D. O. Anderson. 2007. The Multi-Agent Rendezvous Problem. Part 1: The Synchronous Case. *SIAM Journal on Control and Optimization* 46, 6 (2007), 2096–2119. https://doi.org/10.1137/040620552 arXiv:https://doi.org/10.1137/040620552

[22] Matteo Luperto, Mauro Tellaroli, Michele Antonazzi, and Nicola Basilico. 2025. Multi-robot rendezvous in communication-restricted unknown environments via backtracking and semantic frontier-based exploration. *Robotics and Autonomous Systems* 194 (2025), 105137. https://doi.org/10.1016/j.robot.2025.105137

[23] Chase C Murray and Ritwik Raj. 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies* 110 (2020), 368–398.

[24] Gauthier Picard. 2021. Auction-based and distributed optimization approaches for scheduling observations in satellite constellations with exclusive orbit portions. *arXiv preprint arXiv:2106.03548* (2021).

[25] Sameera Ponda, Josh Redding, Han-Lim Choi, Jonathan P How, Matt Vavrina, and John Vian. 2010. Decentralized planning for complex missions with dynamic communication constraints. In *Proceedings of the 2010 American Control Conference*. IEEE, 3998–4003.

[26] Sarvapali D. Ramchurn, Alessandro Farinelli, Kathryn S. Macarthur, and Nicholas R. Jennings. 2010. Decentralized Coordination in RoboCup Rescue. *Comput. J.* 53, 9 (Nov. 2010), 1447–1461. https://doi.org/10.1093/comjnl/bxq022

[27] Arezoo Samiei and Liang Sun. 2024. Distributed Matching-By-Clone Hungarian-Based Algorithm for Task Allocation of Multiagent Systems. *IEEE Transactions on Robotics* 40 (2024), 851–863. https://doi.org/10.1109/TRO.2023.3335656 Conference Name: IEEE Transactions on Robotics.

[28] Shengli Wang, Youjiang Liu, Yongtao Qiu, and Jie Zhou. 2022. Consensus-Based Decentralized Task Allocation for Multi-Agent Systems and Simultaneous Multi-Agent Tasks. *IEEE Robotics and Automation Letters* 7, 4 (Oct. 2022), 12593–12600. https://doi.org/10.1109/LRA.2022.3220155 Conference Name: IEEE Robotics and Automation Letters.

[29] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. 2005. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161, 1 (Jan. 2005), 55–87. https://doi.org/10.1016/j.artint.2004.10.004

[30] Wanqing Zhao, Qinggang Meng, and Paul W. H. Chung. 2016. A Heuristic Distributed Task Allocation Method for Multivehicle Multitask Problems and Its Application to Search and Rescue Scenario. *IEEE Transactions on Cybernetics* 46, 4 (April 2016), 902–915. https://doi.org/10.1109/TCYB.2015.2418052 Conference Name: IEEE Transactions on Cybernetics.

A queue-based approach for fine-tuning the efficiency-fairness tradeoff in distributed satellite scheduling

Shai Krigman
Ariel University
Ariel, Israel
shai.krigman@msmail.ariel.ac.il

Tal Grinshpoun
Ariel University
Ariel, Israel
talgr@ariel.ac.il

Lihi Dery
Ariel University
Ariel, Israel
lihid@ariel.ac.il

ABSTRACT

As the demand for Earth observation satellite (EOS) services grows, allocating limited orbital windows among diverse stakeholders requires balancing overall system utility with equitable access. While distributed constraint optimization (DCOP) has been successfully applied to address user privacy in these settings, existing methods often prioritize efficiency over fairness. A recent hybrid approach offers a compromise between efficiency and fairness. However, it lacks the granularity needed for specific operational contexts. This paper proposes a queue-based distributed algorithm that utilizes multi-level priority queues to resolve resource conflicts. By adjusting the number of queues, the mechanism provides a tunable tradeoff between efficiency and fairness. We provide a formal description of the algorithm and demonstrate its effectiveness in highly-conflicted satellite scheduling scenarios.

KEYWORDS

Earth observation satellites, Distributed Scheduling, Multi-agent systems, Efficiency-fairness tradeoff, DCOP

ACM Reference Format:
Shai Krigman, Tal Grinshpoun, and Lihi Dery. 2026. A queue-based approach for fine-tuning the efficiency-fairness tradeoff in distributed satellite scheduling. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 4 pages.

1 INTRODUCTION

Systems of observation satellites (EOS) [12] are frequently co-financed by multiple stakeholders (e.g., countries, companies, or research institutes) due to their high cost, requiring allocation of shared satellite resources once operational. In multi-satellite settings, scheduling must accommodate requests from multiple users, balancing *efficiency* — maximizing utilization for high-priority tasks — and *fairness*, as perceived by stakeholders [1]. Most existing approaches are centralized, assuming all requests are submitted to a single optimizing authority; however, stakeholders may be reluctant to share sensitive or proprietary information.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

To mitigate these concerns, the distributed constraint optimization problem (DCOP) modeling [6] has been proposed for EOS scheduling [11], followed by several DCOP-based methods [5, 9, 10, 14]. These approaches enable users to coordinate via message exchange and jointly construct a schedule without a central authority.

While DCOP-based algorithms typically emphasize global utility maximization, fairness — namely, equitable resource allocation among stakeholders — remains essential in shared EOS systems.

Prior work introduced a hybrid approach that uses a probabilistic rule to mitigate the "efficiency-first" bias [8]. However, that method provides a static compromise. In this paper, we focus on a queue-based approach that replaces randomization with a structured selection process across multiple priority queues, allowing system operators to explicitly control the efficiency-fairness tradeoff by adjusting the number of queues.

2 PROBLEM DEFINITION

The EOS scheduling problem consists of a set $\mathcal{S}$ of satellites and a set $\mathcal{U}$ of independent users. Each satellite $s \in \mathcal{S}$ follows an orbit plan OP_s and has a capacity limit κ_s, defined as the maximum number of observations during OP_s. Each user $u \in \mathcal{U}$ submits a set $\mathcal{R}$ of observation requests. Each request $r \in \mathcal{R}$ specifies a geographic area to be observed for a duration $\Delta_r \in \mathcal{T}$ within the validity window $[t_r^{start}, t_r^{end}]$, where $t_r^{start}, t_r^{end} \in \mathcal{T}$ and $\mathcal{T}$ denotes the system timeline. The location to be observed is given by p_r (latitude–longitude–altitude).

Given the high cost of Earth observation satellites, they are typically funded by multiple stakeholders with unequal contributions. To reflect these disparities, users are allocated "tokens" representing their entitlement to system usage. Each request r is associated with a reward ρ_r (in tokens), reflecting either the stakeholder's relative contribution or the user's willingness to pay; thus, identical requests may carry different ρ_r values across users.

Each request may induce multiple observation opportunities O. Each opportunity $o \in O$ is characterized by its satellite s_o, start time t_o^{start}, duration Δ_o, and observation location, and yields a reward ρ_o derived from ρ_r and adjusted according to factors such as observation angle and weather conditions. Users generate O based on OP_s, t_r^{start}, t_r^{end}, Δ_r, and p_r. A feasible solution assigns at most one $o \in O$ to each $r \in \mathcal{R}$ such that no two assigned opportunities on the same satellite overlap and the capacity constraint κ_s is satisfied for every $s \in \mathcal{S}$. A desirable solution is both efficient and fair.

3 DCOP MODELING OF THE PROBLEM

A *DCOP* is a tuple $< \mathcal{A}, \mathcal{X}, \alpha, \mathcal{D}, C >$, where $\mathcal{A} = \{A_1, \ldots, A_n\}$ is a finite set of agents; $\mathcal{X} = \{X_1, \ldots, X_m\}$ is a finite set of variables; α :

$\mathcal{X} \to \mathcal{A}$ assigns each variable to a single agent; $\mathcal{D} = \{D_1, \ldots, D_m\}$ is a set of finite domains, where D_i contains the values of X_i; and C is a set of constraints, each $c \in C$ being a function $c : D_{i_1} \times \cdots \times D_{i_k} \to \mathbb{R}_+$ that assigns a non-negative cost to every value combination of its scope. A *complete assignment* assigns values to all variables in $\mathcal{X}$, and an *optimal solution* is a complete assignment of minimal total cost.

Krigman et al. [9] model the EOS scheduling problem as a DCOP in which agents represent users ($\mathcal{A} \coloneqq \{u \in \mathcal{U}\}$), variables represent requests ($\mathcal{X} \coloneqq \{r \in \mathcal{R}\}$), and α assigns each r to its requesting user according to $\mathcal{R}_u$. The domain of X_r is $D_r \coloneqq O_r \cup \{\perp\}$, where O_r is the set of feasible observation opportunities for r and $\perp$ denotes rejection.

Three constraint types are defined. **Unary** constraints assign a cost to each value based on the opportunity reward,[1] **binary** constraints impose ∞ cost on overlapping opportunities assigned to the same satellite and zero otherwise; and **global** constraints over opportunities of each satellite s enforce that the number of assigned opportunities does not exceed κ_s.

4 THE DSARC_xQ ALGORITHM

The *DSA_RC* algorithm [3, 7] is an extension of the classical Distributed Stochastic Algorithm (*DSA*) [13] that can explicitly handle resource capacity constraints. In the EOS scheduling context, each agent in *DSA_RC* computes, for each of its requests, the total number of observation opportunities requested by all agents from the satellite associated with its selected opportunity. If this number exceeds the satellite's capacity, the agent independently decides, for each request, whether to withdraw the corresponding opportunity.

Three approaches were proposed in the literature for making the above decision — an efficiency-aimed approach *DSARC_Eff* [9], a fairness-aimed approach *DSARC_Fair* [3], and a hybrid approach *DSARC_Hyb* [8]. In the next subsection, we propose a novel queue-based approach that enables fine-tuning of the efficiency-fairness tradeoff.

Algorithm 1 presents the *DSA_RC* algorithm in a general manner that encapsulates all the existing approaches together with the new queue-based approach. Each agent distributively executes the algorithm for each request. We focus on the call to the function DecideToRemoveOpp (Line 10), which differntiates between the above approaches. The new queue-based approach is described next.

Algorithm 1: *DSA_RC* run by agent u for request r

Input: p, K

1: $counter \leftarrow 0$
2: $o_r \leftarrow \text{Random}(O_r)$
3: send s_{o_r} and ρ_{o_r} to all agents
4: **while** $counter < K$ **do**
5: $\quad Rew \leftarrow \{\rho_{o_r}\}$
6: $\quad$ collect all agents' s_o and ρ_o
7: $\quad$ **if** $s_{o_r} = s_o$ **then**
8: $\quad\quad Rew \leftarrow Rew \cup \{\rho_o\}$
9: $\quad$ **if** $|Rew| > \kappa_{s_{o_r}}$ **then**
10: $\quad\quad$ **if** DecideToRemoveOpp$(Rew, \kappa_{s_{o_r}})$ **then**
11: $\quad\quad\quad o_r \leftarrow \perp$
12: $\quad\quad\quad$ send o_r to all agents
13: $\quad$ **else if** Random$([0..1)) < p$ **then**
14: $\quad\quad o_r \leftarrow$ valid value with minimal cost in D_r
15: $\quad\quad$ send s_{o_r} and ρ_{o_r} to all agents
16: $\quad counter \leftarrow counter + 1$
17: $X_r \leftarrow o_r$

4.1 Queue-based decision

Our proposed approach is inspired by the biased front-queue selector for web crawling [2].

In this approach, presented in Function 2, the agent partitions all observations associated with the same satellite into x reward-based queues, where x is an input parameter (Line 1). The reward range is divided into x intervals, each mapped to a queue, and each observation is assigned accordingly. The first queue contains the highest-reward observations and the last contains the lowest; the partitioning may be skewed so that, for example, most high-reward observations reside in the first queue. Each queue is sorted in descending reward order (Lines 2–3). A list arr is then constructed (Lines 5–14) as follows: in iteration i, one observation is selected sequentially from each of the first i queues (when non-empty). Thus, the first iteration draws from the first queue only; the second from the first and second queues; the third from the first, second, and third queues; and so on. The process continues until all queues are empty or arr reaches the capacity limit κ_s. Finally, if the agent's opportunity is not included in arr, it is dropped (Lines 15–18).

The construction of arr enables the inclusion of lower-priority opportunities, promoting fairness while maintaining efficiency. Higher-reward opportunities retain a greater likelihood of selection. The parameter x governs the efficiency-fairness tradeoff: $x = 1$ yields the purely efficiency-oriented variant, whereas larger x increases fairness at the expense of efficiency. This algorithm is denoted *DSARC_xQ*, where x is the number of queues.

5 EXPERIMENTAL EVALUATION

We evaluated four decision types: efficiency-oriented (*DSARC_Eff*), fairness-oriented (*DSARC_Fair*), hybrid (*DSARC_Hyb*), and queue-based (*DSARC_xQ*). For *DSARC_xQ*, we tested *DSARC_2Q*, *DSARC_4Q*, and *DSARC_8Q*. Results were compared to centralized post-processing (*DSA_PP*) [9]. Parameters were set to $p = 0.7$ and $K = 10$ in all algorithms, following [9].

Problem instances were based on the "highly-conflicted problems" of [10]: a 10-minute horizon, three satellites (capacity 20 each), ten users submitting $|\mathcal{R}_u| = \{2, 4, \ldots, 20\}$ requests, and ten opportunities per request. Request validity windows were $[100 : 200]$, with $\Delta_o = [10 : 20]$ and $\rho_o = [10 : 50]$. We also generated a second set of high-density problems.

To capture diverse settings, we considered four reward distributions per instance: uniform, exponential, normal, and a bimodal split (two user groups with high vs. low rewards).

[1] Since DCOPs are formulated as minimization problems with non-negative costs, the cost of opportunity o is defined as $cost_o = maxCost - \rho_o$, where $maxCost > \max_{o \in O} \rho_o$. The cost of $\perp$ is $maxCost$.

Function 2: Queue-based DecideToRemoveOpp

Input: Rew, κ_s, x

```
Divide Rew into x queues according to ρ_o
for all queues do
    sort queue from highest ρ_o to lowest
cyc ← 0, arr ← ∅
while queues are not empty do
    for i ← 0 to cyc − 1 do
        if queue[i] is not empty then
            move next o from queue[i] to arr
            if |arr| = κ_s then
                break while
    if cyc = x then
        cyc ← 0
    else
        cyc ← cyc + 1
if o_r ∈ arr then
    return false
else
    return true
```

Efficiency was measured by the total number of scheduled requests and total rewards. Fairness was evaluated via the Gini coefficient [4], computed over total rewards and over scheduled requests (with $\rho_o = 1$).

Results for the highly-conflicted set appear in Figure 1. For small workloads, all methods schedule all requests; beyond capacity, scheduling decreases accordingly. Results are therefore reported from 48 requests onward.

Efficiency and fairness exhibit a clear tradeoff: gains in one reduce the other. *DSARC_Hyb* provides a balanced but fixed compromise. When no prior knowledge is available, *DSARC_Hyb* is a robust choice. With distributional insight, *DSARC_xQ* can perform better. Under uniform rewards, *DSARC_2Q* exceeds *DSARC_Hyb* in total rewards while matching its fairness (Gini). Under exponential rewards, *DSARC_2Q* matches *DSARC_Eff* in efficiency and approaches *DSARC_Hyb* in fairness. For bimodal rewards, *DSARC_2Q* and *DSARC_4Q* achieve the highest fairness in scheduled requests.

6 CONCLUSION

We have presented the *DSARC_xQ* algorithm for managing the efficiency-fairness tradeoff in distributed satellite scheduling. By implementing a structured priority queue system, this approach moves beyond the static compromise of the previous hybrid coordination model, offering a tunable solution that can be tailored to specific reward distributions. Our evaluation demonstrates that while the foundation of distributed scheduling relies on efficient DCOP modeling, the addition of multi-level queues effectively prevents the systemic exclusion of users with lower-priority tasks without requiring centralized coordination.

Possible future research directions include investigating the application of this mechanism in dynamic environments and adopting advanced privacy-preserving protocols to further secure stakeholder information.

ACKNOWLEDGEMENTS

This research was supported by the Ministry of Innovation, Science & Technology, Israel.

REFERENCES

[1] Nicolas Bataille, Michel Lemaitre, and Gerard Verfaillie. 1999. Efficiency and Fairness when Sharing the Use of a Satellite. In *Artificial Intelligence, Robotics and Automation in Space*, Vol. 440. 465.

[2] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. 1998. Efficient crawling through URL ordering. *Computer networks and ISDN systems* 30, 1-7 (1998), 161–172.

[3] Lihi Dery, Tal Grinshpoun, and Ilya Khakhiashvili. 2025. Distributed Course Allocation with Asymmetric Friendships. *Autonomous Agents and Multi-Agent Systems* 39, 1 (2025), 26.

[4] Corrado Gini. 1936. On the measure of concentration with special reference to income and statistics. *Colorado College Publication, General Series* 208, 1 (1936), 73–79.

[5] Ryan Harrod, Shreya Parjan, and Steve Chien. 2022. Distributed observation allocation for a large-scale constellation. (2022).

[6] Katsutoshi Hirayama and Makoto Yokoo. 1997. Distributed partial constraint satisfaction problem. In *International conference on principles and practice of constraint programming*. Springer, 222–236.

[7] Ilya Khakhiashvili, Tal Grinshpoun, and Lihi Dery. 2021. Course Allocation with Friendships as an Asymmetric Distributed Constraint Optimization Problem. In *2021 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. IEEE, 688–693.

[8] Shai Krigman, Lihi Dery, and Grinshpoun. 2025. A Fair Share: Fair Allocation of Satellite Observation Windows According to User Preferences in a Distributed Setting. In *Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct '25)*. 245–248.

[9] Shai Krigman, Tal Grinshpoun, and Lihi Dery. 2024. Scheduling of Earth observing satellites using distributed constraint optimization. *Journal of Scheduling* 27, 5 (2024), 507–524.

[10] Gauthier Picard. 2021. Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions. *arXiv preprint arXiv:2106.03548* (2021).

[11] Gauthier Picard, Clément Caron, Jean-Loup Farges, Jonathan Guerra, Cédric Pralet, and Stéphanie Roussel. 2021. Autonomous agents and multiagent systems challenges in Earth observation satellite constellations. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*. 39–44.

[12] Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. 2020. Agile Earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. *IEEE Systems Journal* 15, 3 (2020), 3881–3892.

[13] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. 2005. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161, 1-2 (2005), 55–87.

[14] Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. 2025. Decentralized, decomposition-based observation scheduling for a large-scale satellite constellation. *Journal of Artificial Intelligence Research* 82 (2025), 169–208.

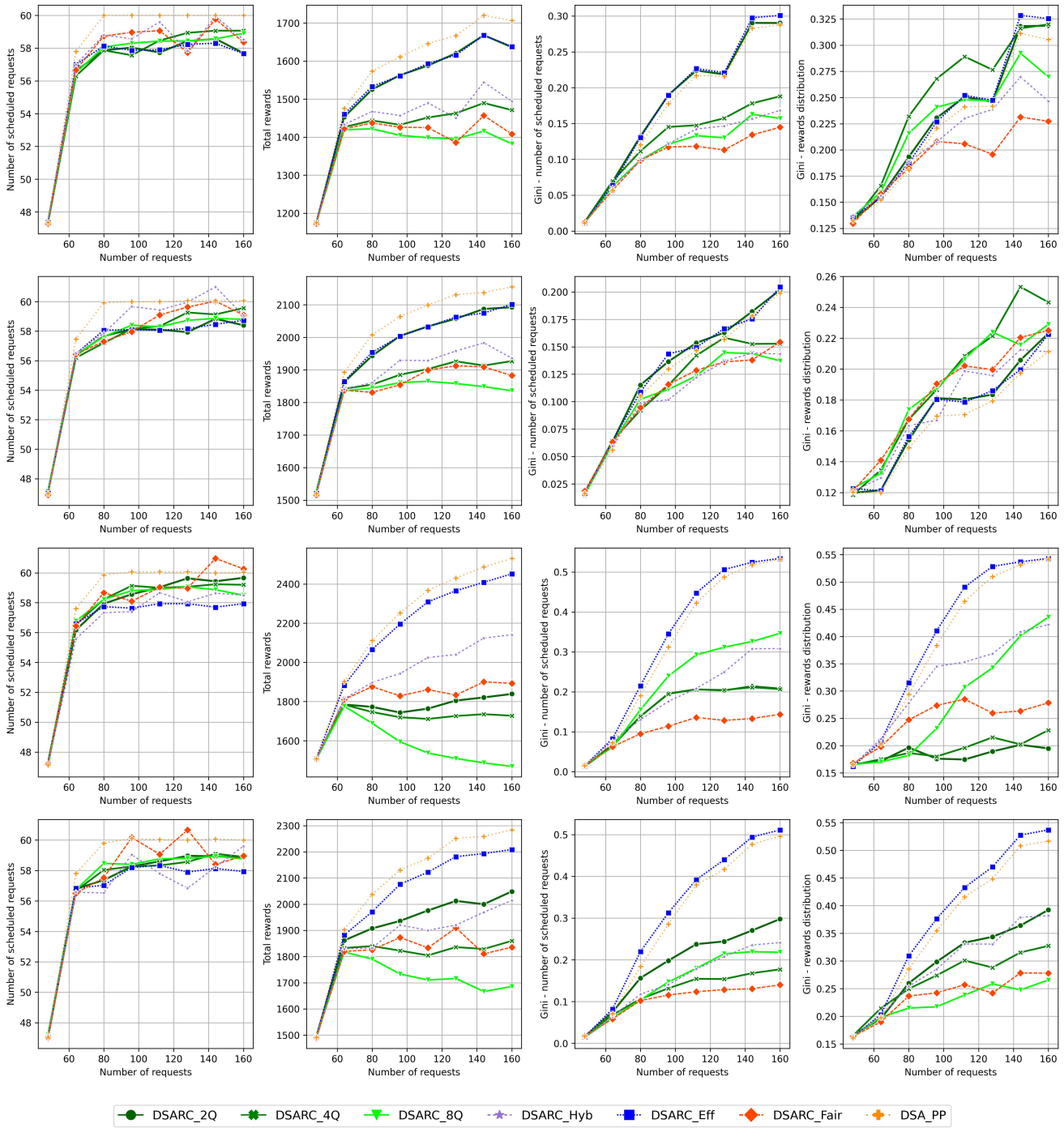


Figure 1: Comparison of scheduling algorithms. Each row corresponds to a different reward distribution: Uniform (first row), Exponential (second row), Normal (third row), and Two equal groups (fourth row). The columns represent the following evaluation metrics: (1) Number of Scheduled Requests, (2) Total Rewards, (3) Gini Coefficient for Scheduled Requests, and (4) Gini Coefficient for Total Rewards.

Onboard Planning and Scheduling for Autonomous Satellite Operations Towards Multi-Agent Cooperation

Riccardo Maderna
AIKO Srl
Turin, Italy
riccardo.maderna@aikospace.com

Marco Morgese
AIKO Srl
Turin, Italy
marco.morgese@aikospace.com

ABSTRACT

This paper presents an onboard planning and scheduling pipeline designed to enable autonomous satellite operations. We first provide a system-level overview of the architecture, highlighting how the planning agent ingests state and contextual information and issues time- and resource-constrained commands in real time. We then discuss how interaction with other onboard applications - payload data processing, health monitoring, orbit maintenance, and cooperative tasking - can enable the highest degree of operational autonomy. The behavior of the autonomous agent is exemplified through a set of simulation scenarios that illustrate its ability to adapt to varying mission conditions, evolving operational constraints, and off-nominal events. Finally, we introduce a concept for managing multi-agent coordination, focusing on task assignment. This work shows a concrete implementation of onboard satellite autonomy which enables more resilient, scalable, and efficient autonomous operations in current and future space missions.

KEYWORDS

Autonomous spacecraft, Mission operations, Planning and scheduling, Multi-agent cooperation, Task assignment

ACM Reference Format:
Riccardo Maderna and Marco Morgese. 2026. Onboard Planning and Scheduling for Autonomous Satellite Operations Towards Multi-Agent Cooperation. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 6 pages.

1 INTRODUCTION

The growing commercial competition in space is driving the need for more efficient and scalable missions. Advanced autonomy is emerging as a key enabler, supporting the management of increasingly complex space systems, improving mission output, cutting operational costs, and enabling new mission concepts [14]. Enhanced onboard processing now allows intelligence to be embedded directly in spacecraft, and the tight integration of onboard and ground operations improves responsiveness, especially for user-driven missions such as Earth observation. This integration enables just in time tasking and continuous schedule optimization, making the system more adaptable and responsive to dynamic requirements. Onboard autonomy also unlocks event driven mission concepts that depend on unpredictable phenomena.

Traditional onboard planning has focused mainly on reactive repair of ground generated schedules, as seen in systems like VAMOS [15] , but purely reactive approaches [4, 7] can be short-sighted and suboptimal. Conversely, fully proactive planning offers long term reasoning but lacks flexibility when unexpected events occur, usually requiring full re-plan [1]. The most effective strategy is therefore a hybrid approach that combines long-term planning with reactive adjustments. This work describes an integrated onboard planning and scheduling pipeline enabling full goal-oriented autonomy. A proactive component generates a long horizon flexible plan, while a reactive component executes it, absorbs uncertainties, and refines actions using real time information. A reasoning engine monitors plan validity and triggers re-planning when needed. This system combines the timeline formalism [3] and conditional procedures, providing an effective framework to manage concurrent activities, constraints, resources and events, and it is designed for broad applicability across missions.

As autonomy scales to constellations, the ground segment primarily takes on the role of allocating user requests among satellites with different capabilities, while each spacecraft autonomously plans and executes its assigned tasks. This aspect is explored in Section 5 of the paper.

2 ONBOARD PLANNING AND SCHEDULING SYSTEM

The architecture of the planning and scheduling agent (Figure 1) is composed of three primary components: Request Manager, Planning Manager, and Execution Manager. The Request Manager is responsible for managing incoming user requests. It monitors the life cycle of each request from submission to completion and assigns priority levels based on mission rules and operational constraints to ensure that the system processes requests in an optimal order.

The Planning Manager provides the proactive component: it generates long-term flexible plans of activity using a timeline-based approach. It incorporates user requests, mission goals, operational constraints, and mandatory activities received from operators. The resulting flexible plan can be seen as an envelope of deterministic schedules, each one representing a fixed-in-time sequence of actions or commands. This flexibility allows for absorbing uncertainties during execution without the need for re-planning, which strongly reduces the computational load of the whole planning pipeline and increases the predictability of the satellite's behavior.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

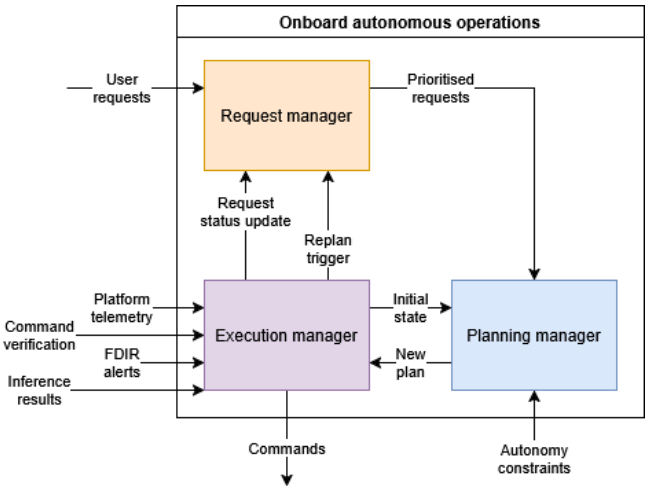


Figure 1: High-level system architecture.

The Execution Manager translates planned activities into a sequence of commands via conditional procedures. Flexibility of the plan and conditional branches are exploited to optimize the mission return and to adjust the schedule in response to just-in-time information (like edge data processing results) and contingencies. The Execution Manager also monitors the validity of the plan and triggers a re-planning process in case of: violation of resource availability envelopes, indicating a mismatch between forecast and actual resource use; occurrence of anomalies that invalidate the plan; reception of new information, including new user requests or mandatory activities from ground operators.

Both Planning and Execution Managers implement generic engines that are highly configurable to mission requirements ensuring high re-usability.

2.1 Proactive planning

The Planning Manager is based on timeline-based planning [3], a formalism well suited for mission planning thanks to its expressiveness and natural alignment with how space mission constraints and resources are modeled. In this approach, the world is represented through state variables whose values evolve over time. These variables may be external (e.g., ground station visibility) or internal (e.g., spacecraft orientation). Their evolution is encoded in timelines, which are sequences of tokens representing flexible time intervals during which each variable holds a specific value. A chronicle aggregates all timelines along with constraints and resource information.

The planner searches in the space of chronicles, starting from a partial one and iteratively resolving flaws (i.e., constraint violations) until a complete, feasible solution is found. The resulting plan consists of timelines and their mutual constraints. Since constraints are only inserted to ensure feasibility, the obtained plan maintains margins for flexibility that is exploited at execution time. To handle both controllable and uncontrollable temporal relations, each chronicle maintains a Simple Temporal Network with Uncertainty (STNU), which models uncertain durations, precedence constraints, and synchronizations with external events. The STNU must remain consistent and dynamically controllable to ensure executable plans [6]. Resource feasibility is also maintained through dedicated managers for binary, capacity, and consumable resources. A pseudo-code overview of the planning routine is the following:

Algorithm 1 Plan(initial_state, final_state, goal_list

```
1:  function PLAN(initial_state, final_state, goals)
2:      chronicle ← INITIALIZE(initial_state, final_state)
3:      exploration_stack ← ∅
4:      while chronicle is flawed do
5:          flaw ← GETHIGHESTPRIORITYFLAW(chronicle, goals)
6:          resolvers ← COMPUTERESOLVERS(flaw)
7:          if |resolvers| > 1 then
8:              ADDDECISIONPOINT(exploration_stack)
9:              resolver ← GETBESTRESOLVER(resolvers)
10:             ADDRESOLVER(exploration_stack, resolver)
11:         end if
12:         APPLY(chronicle, resolver)
13:         if chronicle is inconsistent then
14:             backtrack to the previous decision point
15:         end if
16:     end while
17:     return EXTRACTPLAN(chronicle)
18: end function
```

The planner supports two goal types: permanent and opportunity goals. Permanent goals cover things like keeping batteries charged or maximizing data acquisition and can adapt to changes in mission priorities. Opportunity goals refer to time bound tasks, such as targeted imaging requests, which are provided as a prioritized list by the Request Manager, and the planner selects the best subset that can coexist with other activities while preserving feasibility.

2.2 Reactive execution loop

The Execution Manager implements reactivity at two levels. First, it executes the flexible plan by assigning fixed times at time-points in the STNU. Firings of controllable time-points are chosen to optimize mission return; uncontrollable ones are based on sensed events or the conclusion of procedures.

The second level is the implementation of activities via conditional procedures. It is supported by a procedure execution engine that interprets procedures written in a compact domain-specific language. The engine processes statements sequentially - evaluating expressions, assignments, control flow, and command statements - until it encounters a wait condition that is not satisfied. When state changes and the wait condition is satisfied, execution resumes. Conditional branches and wait conditions can be evaluated based on internal procedure variables and mission state parameters, like satellite telemetry or the state of dispatched commands.

3 INTERACTIONS WITH ONBOARD APPLICATIONS

Onboard planning beats ground mission planning as it can exploit real-time information. Therefore, it becomes more powerful in combination with other onboard applications that can provide the same inputs ground operations teams use for mission planning, but

with low latency. By doing so, it is possible not only to increase mission efficiency but to unlock new, highly dynamic, mission concepts. Specifically, four classes of supporting applications are considered in this paper:

- **Onboard payload data processing:** applications able to process payload data to produce actionable information on board, like event detections. Such insights are exploited by the autonomous operations agent in two ways: they are input to evaluate conditional branches of procedure (for instance, discarding low quality data and abort further processing) and they are used to generate requests onboard (for instance, follow-up observations following fire detection).
- **Spacecraft health monitoring:** applications able to detect and classify anomalies, propose recovery or mitigating actions, and monitor the effect of the recovery plan execution [9, 13]. It produces two types of data: i) alerts that inform the autonomous agent to avoid usage of unavailable resources or operating modes; ii) recovery plans that are inserted in the plan as autonomy constraints.
- **Autonomous orbit maintenance:** applications able to compute station keeping and/or collision avoidance maneuvers [2, 5, 12], which are autonomy constraints for the autonomous operations agent.
- **Cooperative tasking:** application that enables distributed and cooperative tasking among satellites in a constellation endowed with inter-satellite links. It provides a source of new user requests and a way to re-distribute requests that the satellite is unable to satisfy efficiently.

4 EXAMPLES OF AGENT BEHAVIOR

An example scenario illustrates the algorithm's behavior and its advantages over common satellite planning methods. The case concerns a small Earth observation mission whose primary goal is acquiring user requested targets, while secondary wide area acquisitions increase mission return. Onboard data processing (e.g., cloud detection) helps adjust acquisition frequency to save resources.

The planning problem includes two external timelines - eclipses and ground station visibility - and two internal timelines – operating mode (feasible states: idle, observation, monitoring, downlink, ground control) and spacecraft orientation (feasible states: sun pointing, antenna to ground, camera to ground, slewing). Battery level and memory usage are treated as consumable resources. User requests are modeled as opportunity goals; extra data takes as a permanent goal. At execution level, conditional procedures describe that the observation activity can switch between high and low frequency acquisition depending on local conditions, while monitoring can escalate to observation when beneficial.

Figure 2 shows an example of the planning process. Initially (Fig. 2a), the external timelines are fully defined (in blue), while internal ones contain only initial states and mandatory maintenance windows. Red tokens indicate unsupported elements. Flexible time intervals appear in lighter colors. A worst-case battery profile is shown at the bottom. Fig. 2b shows an intermediate state after evaluating opportunity goals: some are excluded due to conflicts, resource limits, or inconsistencies. Fig. 2c presents the output plan, with all tokens supported and remaining flexibility, which depends on scenario uncertainty (e.g., activities with uncontrollable durations).

Figure 3 highlights reactive execution behavior. The first row shows an opportunity goal, leading to a planned observation over an area with some flexibility before switching to monitoring. Expected and actual battery profiles are shown with dashed and solid lines. The bottom row shows the reactive loop's refinements. In Fig. 3a, ongoing observation is executed at high frequency. The mode adjusts dynamically based on cloud coverage until the opportunity goal ends (Fig. 3b). Low frequency acquisitions consume less battery, generating a surplus; the execution loop therefore extends observation to pursue permanent goals (Fig. 3c). When battery or memory exceeds thresholds, a new local goal triggers a switch to the next activity (Fig. 3d).

5 MULTI-AGENT COOPERATION

Constellations are a use case of particular interest for autonomous satellites as they allow ground segments to operate a large number of spacecraft at a fraction of the effort. No detailed planning is performed on the ground, which must only distribute user and onboard generated service requests. This section describes a multi-agent negotiation system concept for the task distribution problem.

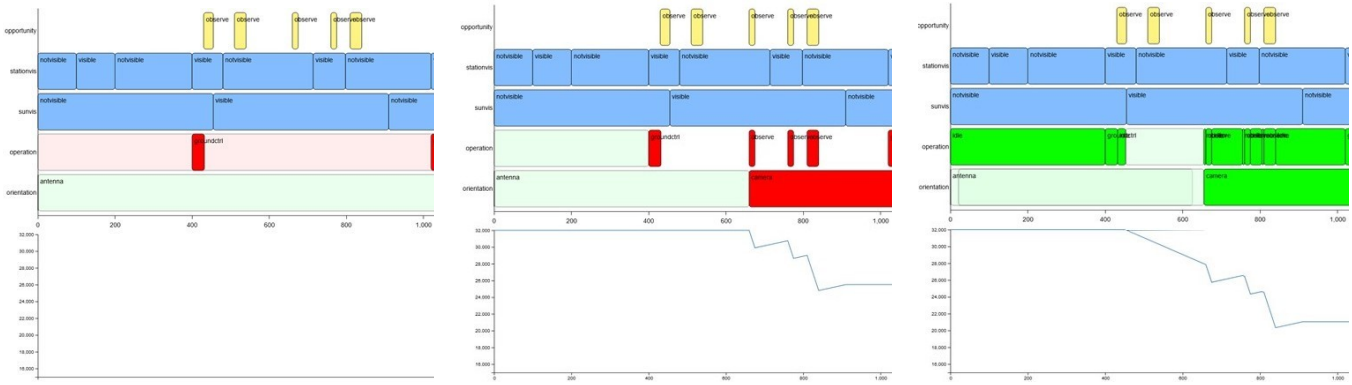
The ground system collects user and onboard generated service requests. When a satellite comes into contact with a ground station, a subset of the pending requests is assigned to it, depending on the satellite's capabilities and availability. The problem can be modeled as an auction with the ground segment as the auctioneer and satellites in the constellation as negotiating agents. The challenge is that satellites contact the ground segment at different times, making it impossible for all agents to place their bids. Therefore, the ground segment maintains a proxy representation of agents, which participate in auctions. Every time a satellite communicates with the ground, the state of the corresponding proxy is updated, and an auction for pending tasks is run among all proxies. Figure 4 gives a pictorial view of the system architecture, whereas the sequence diagram in Figure 5 describes the auction process triggered by a new satellite contact.

Each proxy is a tuple $p_i = (R_i, L_i, K_i, \Pi_i, \mathcal{K}_i, \Delta_i, \gamma_i)$ where:

- R_i is the set of assigned pending requests,
- L_i is the set of resource level arrays indicating the predicted resources levels for the various satellite resources at future time instants,
- K_i is the set of services that satellite can currently provide,
- Π_i is the set of payloads available onboard the satellite to provide services,
- $\mathcal{K}_i : K_i \rightarrow \mathcal{P}(\Pi_i)$ is a function that maps each service $k \in K_i$ to the subset of payloads required to perform the service,
- $\Delta_i : K_i \rightarrow \mathbb{R}$ is a function that maps each service $k \in K_i$ to the amount of time required to perform the service,
- $\Gamma_i : K_i \rightarrow \mathbb{R}^2$ is a function that maps each service $k \in K_i$ to the amount of resources used to perform the service.

The main steps of the process are:

(1) **Proxies update**. The state of the Proxy of the communicating satellite is updated with real data, while the other Proxies are updated through simulation according to their expected evolution.



(a) Initial Chronicle. (b) Intermediate status. (c) Final flexible plan.

Figure 2: Example of proactive planning process from initial Chronicle (portion) to final plan. The top row (yellow) shows pending requests to plan for; blue marks external timelines; red and green mark unsupported and supported internal tokens, respectively. Flexible time intervals appear in lighter colors. The bottom graph describes the worst-case battery profile.

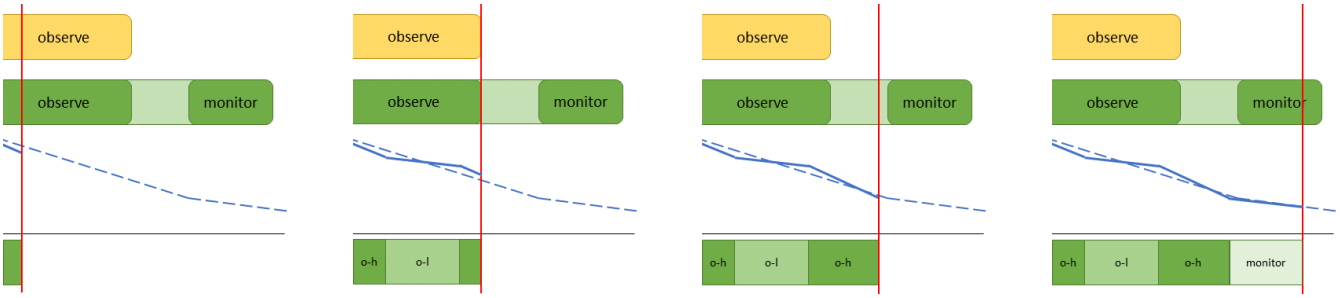


Figure 3: Example of reactive execution process. An observation request (yellow) has been inserted in the plan (green). (a) Ongoing observation is executed with high frequency acquisitions; (b) the mode adjusts dynamically based on cloud coverage; (c) plan flexibility allow to continue high-frequency observations to exploit the battery surplus; (d) when battery lowers, a new local goal triggers the next activity.

(2) **Auction**. The Auctioneer broadcasts a set R of pending requests (default strategy for selection is FIFO, but it can be tailored on mission requirements) and Proxies answer with a bid for each request. The Auctioneer ranks the bids and assigns the highest-valued requests to the winning Proxies, which updates their state. The number of announced and assigned requests is configurable, and this step can be performed multiple times to assign all requests.

(3) **Request transmission**. Once the assignment process has been completed, a message is sent to the communicating satellite before the end of the visibility window, containing the new requests assigned to it since the last contact.

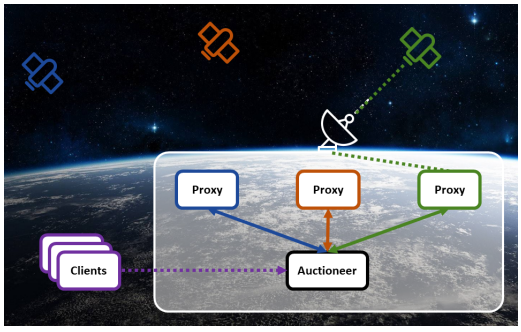


Figure 4: Pictorial view of negotiation system.

5.1 Bid generation

Upon reception of an announcement, each Proxy generates a bid for every announced request, which expresses the capability of the satellite to satisfy the request timely and efficiently. Bids are computed independently for each request in the announcement.

The Proxy p_j does not bid if the satellite is unable to provide the requested service or it has no visibility on the target before the request expiration date; otherwise, it bids a convex combination of N_b bidding terms $\gamma_n \in [0, 1]$: $b_j = \sum_{n=1}^{N_b} w_n \gamma_n$

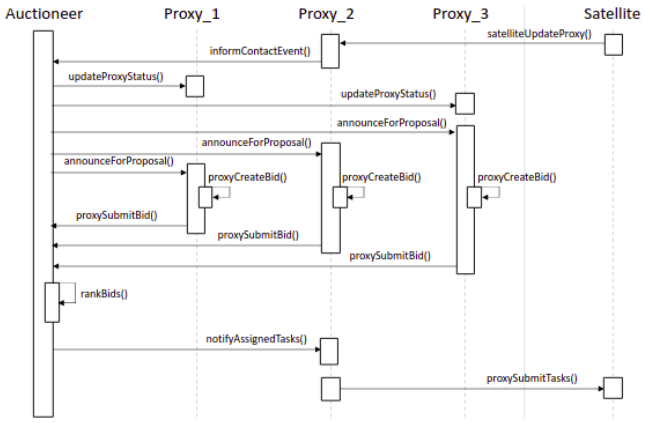


Figure 5: Auction process triggered by a new satellite contact.

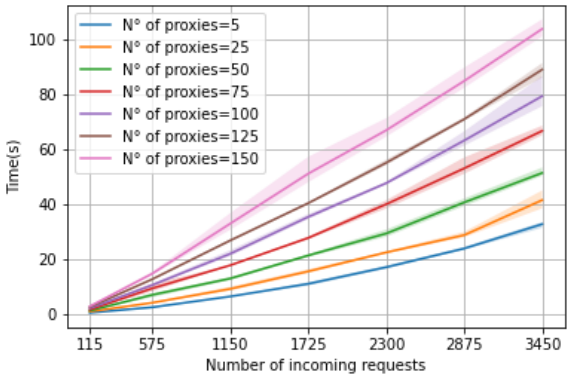


Figure 6: Auction time for single-item auctions.

In this first proof of concept the considered bidding terms are:

- **Assigned requests:** the bid decreases with the number of already assigned requests.
- **Request priority:** the bid increases with the priority and the approaching expiring date t^{exp} of the request. This term is independent of the Proxy status but serves as an incentive to assign high priority requests first.
- **Request satisfaction time:** the bid is higher the shorter the time required for the satellite to execute the requested service and transmit the data back to ground.
- **Satellite availability:** when conflicting requests are assigned to the same satellite, it results in a delay in the satisfaction of all but the highest priority request. Therefore, the higher the number and priority of conflicting requests the lower the bid.
- **Satellite resources:** the bid is lower the less resource availability of the satellite.

5.2 Preliminary analysis

The reference scenario for preliminary analysis was an Earth Observation constellation in LEO with no inter-satellite links. Satellites in the constellation are equipped with optical, SAR or both payloads, with a total of four different services that can be provided (optical image, SAR spot, SAR strip, optical+SAR combined acquisition). Simulations have been run with random initialization on an off-the-shelf laptop with Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz and 12 GB of RAM.

Time complexity analysis was performed for increasing constellation size (up to 150 satellites, with uniform distribution of satellite type) and user requests per episode (up to 3450, with distribution of service type inversely proportional to resource requirements). Figure 6 shows the results for single-item auctions demonstrating the feasibility of running auctions well within the duration of a ground contact window. Experiments have been also conducted by changing the announcement size and assignment size: larger announcement size allows agents to bid for the best requests first for increased computational time, larger assignment size reduces the number of auction rounds.

Sensitivity analysis has been conducted to evaluate the behavior against changing weights of the bid terms. Considered cases were: uniform relative weight, ablation cases with one null weight, opposite cases with one weight set to a high value. Analysis explored impact on load balance (variance in number of tasks assigned to satellites), the number of conflicting tasks assigned to each satellite, and bid evolution over the auction rounds (both in terms of overall value and for each bid term).

6 CONCLUSION

The present work described a proactive-reactive approach for onboard automated planning. The solution is highly flexible and can be applied to a wide variety of missions. The proactive loop allows for long-term, goal-oriented planning while the reactive loop enables local decision-making and plan refinement in view of uncertainty and contingencies. Parts of the pipeline have been tested in testbed (Tyvak International, D-Orbit) and in-orbit demonstrations between 2023-2026 (D-Orbit [10], AI-Express project [8, 11]), whereas a demonstration of the complete agent is expected in 2027 (In-Orbit Space Lab project, others TBC).

Extension to the multi-agent scenario requires further study to address two main aspects. First, the impact of deferred proxy updates on task assignment performance, which becomes even more critical when satellite-to-satellite communication is enabled. Second, the interaction between the planning agent and the onboard cooperative tasking application, which have been for now treated separately.

REFERENCES

[1] M. Troesch et al. 2020. MEXEC: An Onboard Integrated Planning and Execution Approach for Spacecraft Commanding. In *International Conference on Automated Planning and Scheduling (ICAPS 2020)*. 9.
[2] S. Nag et al. 2021. Prototyping operational autonomy for space traffic management. *Acta Astronautica* 180 (2021), 489–506.
[3] M. Ghallab, D. Nau, and P. Traverso. 2016. *Deliberation with Temporal Models*. Cambridge University Press. 114—-154 pages.
[4] E. Herz, D. George, T. Esposito, and K. Center. 2014. Onboard Autonomous Planning System. In *SpaceOps 2014 Conference*.
[5] J. Kruger, S. S. Hwang, and S. D'Amico. 2024. Starling Formation-Flying Optical Experiment: Initial Operations and Flight Results. *preprint arXiv:2406.06748* (2024).
[6] P. Morris, N. Muscettola, and T. Vidal. 2001. Dynamic control of plans with temporal uncertainty.

[7] C. Powell and A. Riccardi. 2022. On-board re-planning of an earth observation satellite for maximisation of observation campaign goals. In *73rd International Astronautical Congress (IAC 2022)*. 11.
[8] AIKO Srl. [n.d.]. AIX - Smart In-Orbit data processing. https://www.aikospace.com/blog/aix-smart-in-orbit-data-processing [Accessed: 2026-04-10].
[9] AIKO Srl. [n.d.]. gifted_GENE. https://aikospace.com/products/gifted-gene [Accessed: 2026-04-10].
[10] AIKO Srl, D-Orbit Spa, and Unibap. [n.d.]. Successful completion of the In-Orbit Demonstration campaign for orbital_OLIVER.
[11] Planetek Italia Srl. [n.d.]. Third AI-eXpress satellite in orbit for a new era of edge intelligence in space. https://www.planetek.it/en/news_events/news_archive/2025/11/third_ai_express_satellite_in_orbit_for_a_new_era_of_edge_intelligence_in_space [Accessed: 2026-04-10].
[12] F. Toussaint, J. Thomassin, and S. Laurens. 2022. ASTERIA in-orbit testing on OPSSAT: an on-board autonomous orbit control solution including collision risks avoidance.
[13] A. Wander and R. Förstner. 2013. Innovative Fault Detection, Isolation and Recovery Strategies On-Board Spacecraft: State of the Art and Research Challenges. *Deutsche Gesellschaft für Luft- und Raumfahrt* (2013).
[14] D. Wang, J. A. Russino, C. Basich, and S. Chien. 2022. Analyzing the Efficacy of Flexible Execution, Replanning, and Plan Optimization for a Planetary Lander. In *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling*, Vol. 32. 9.
[15] M. T. Wörle and C. Lenzen. 2013. Ground Assisted Onboard Planning Autonomy with VAMOS.

The Role of Central Nodes in Multi-Task Allocation for Distributed Space-Based Observation Systems

Vincenzo Messina
Technical University of Munich
Ottobrunn, Germany
vincenzo.messina@tum.de

Marco Camporeale
Technical University of Munich
Ottobrunn, Germany
m.camporeale@tum.de

Rob Vingerhoeds
Fédération ONERA - ISAE-SUPAERO - ENAC, Université de Toulouse
Toulouse, France
rob.vingerhoeds@isae-supaero.fr

Alessandro Golkar
Technical University of Munich
Ottobrunn, Germany
golkar@tum.de

ABSTRACT

This work presents a reward-based selective propagation algorithm that enables distributed agents to evaluate, exchange, and bid task-specific reward functions while minimizing communication overhead. Distributed observation satellites in a Walker-Delta constellation exchange multiple tasks of observing target objects in Sun-Synchronous Orbits. We analyze the impact of the number of satellites, the number of target objects, and the fraction of central nodes on the task's diffusion, task performance quality, and bidding status over time. We investigate constellations with up to 5,000 satellites and 500 concurrent target objects, with a fraction of the central nodes ranging from 1% to 100% of the total number of satellites. Results show that the proposed framework maintains high performance and fast convergence even under heavy task loads, with convergence times below 100 s for networks with as few as 5% central nodes. Increasing the number of satellites reduces the required central node fractions to achieve faster convergence, suggesting that a minimum absolute number of central nodes is required to achieve maximum network responsiveness for the given constellation geometry. We identified a power-law relationship among network size, central node fraction, and link efficiency, offering valuable insights into architectural trade-offs for future Distributed Satellite Systems.

KEYWORDS

Distributed Satellite Systems, Task Allocation, Decentralized Networks, Large-Scale Simulation, Object Detection

ACM Reference Format:

Vincenzo Messina, Marco Camporeale, Rob Vingerhoeds, and Alessandro Golkar. 2026. The Role of Central Nodes in Multi-Task Allocation for Distributed Space-Based Observation Systems. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 9 pages.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

1 INTRODUCTION AND BACKGROUND

Distributed Satellite Systems (DSS) are increasingly adopted for earth observation missions due to their inherent advantages in scalability, resilience, and spatial coverage. As mission objectives expand to include the simultaneous observation of large numbers of dynamic target objects, efficiently allocating tasks across DSS has emerged as a key systems engineering challenge [2].

This paper investigates the impact of central nodes on the performance of distributed multi-task allocation in large-scale DSS. We extend the decentralized task allocation framework, presented in [8], to a multi-task reward-based selective propagation framework in which satellites exchange compact task-level reward metrics, represented by the Reward Function (RF), instead of full state information, enabling distributed bidding under intermittent communication and without prior knowledge of other agents' capabilities. We refer to central nodes as satellites that act as hubs for task coordination and data propagation; task propagation occurs only when at least one of the communicating satellites is a central node. We consider a Walker-Delta Constellation with up to 5,000 observer satellites that need to observe up to 500 objects in space, considered as target objects that represent debris, other satellites, or natural bodies, distributed along Sun-synchronous orbits (SSO). The results focus on network performance, represented by the evolution of the maximum reward, the bidding status over time, and the diffusion of task knowledge. We analyze different levels of centralization by varying the fraction of central nodes relative to the total number of satellites. We observe a power law among network size, the fraction of central nodes, and the link efficiency, defined as the ratio of the total number of links n_L before convergence to the number of tasks n_T. The term DSS refers to satellite systems comprising two or more spacecraft that communicate to accomplish the mission goal [7]. They dynamically form networks of heterogeneous satellites designed to reduce revisit times, cover large areas at higher resolution, or minimize data access latency. This is essential for real-time Earth Observation missions, for monitoring deforestation, studying sea-ice coverage, tracking agricultural evolution and weather impacts, as well as for humanitarian aid, governments, and NGOs to monitor oceans and conflict zones [1]. Centralized architectures are

vulnerable to single points of failure and are fundamentally incompatible with federated systems; they are not beneficial for resource allocation in planning and scheduling when the number of satellites increases [2]. Different examples of DSS are implemented nowadays, such as satellite constellations for communication, such as Starlink and OneWeb, or for Earth Observation, such as PlanetLabs [1]. Many studies focus on the importance of satellite collaborations. Dynamic target (DT) is an example that highlights the importance of DSS, in which look-ahead sensor data is acquired, rapidly analyzed, and used to drive subsequent observations by the same spacecraft or another spacecraft in a trail. Use cases for such a capability include: cloud avoidance [4], storm hunting, search for planetary boundary layer events [14], plume study, and beyond [3]. Federated Satellite Systems (FSS) extend this paradigm further, enabling satellites from different operators to autonomously share resources through standardized protocols [11]. This improves mission efficiency through resource pooling and enhanced resilience [9], while reducing barriers to entry for smaller operators.

Schaefer et al. [12] demonstrate that inter-satellite collaboration reduces revisit time and increases sample count for wildfire-observing missions, highlighting the utility of coordinated DSS operations. Existing methods have already been developed and investigated in the field of DSS task scheduling.

Parjan et al.[10] propose a broadcast-based coordination methodology (BD) in which satellites announce their task selections to the network without requiring detailed knowledge of other agents' capabilities. In this approach, each satellite independently evaluates its suitability for available tasks using local heuristics, then broadcasts status information, such as task-satisfaction confirmations or priority indicators. These methods achieve near-centralized performance but face communication bandwidth challenges as the network size grows and assume that all satellites know the tasks at the problem's initialization.

Zilberstein et al. [15] extend the BD method to formulate satellite scheduling problems as distributed constraint optimization problems (DCOPs). DCOP frameworks use Geometric Neighborhood Decomposition heuristics to subdivide the global problem into manageable subproblems. This approach reduces messaging overhead by limiting coordination to satellites within smaller neighborhoods, whose boundaries are determined by orbital mechanics and communication capabilities. After the subdivision, Neighborhood Stochastic Search (NSS), a method built on the earlier BD, solves these subproblems. DCOP demonstrates improved scalability compared to fully centralized approaches and BD, particularly in dense task environments where decomposition naturally aligns with spatial task clustering. They assume agents possess knowledge of other agents' capabilities and that all agents have immediate knowledge of all tasks. Many DCOP solvers require significant computational resources and extensive inter-agent communication, potentially exceeding the constraints of resource-limited satellites.

Li et al. [6] propose an extension of the Consensus-Based Bundle Algorithm (m-CBBA), where individual satellites determine the full task schedule locally and achieve mutual consensus through iterative communication. An extension of this algorithm is the Asynchronous m-CBBA (m-ACBBA), in which satellites predict communication window opportunities and account for their possible asynchrony. These methods are generally flexible and can guarantee highly-optimal solutions; however, the computational and communication overheads do not scale efficiently with the number of satellites and tasks. Table 1 summarizes the discussed frameworks, highlighting the strengths and limitations identified in each. While decentralized satellite task allocation methods offer clear benefits, they also entail inherent trade-offs, particularly in scalability and communication overhead as the number of satellites and tasks grows.

This study investigates how the fraction of central nodes f_{CN} affects convergence time, task knowledge diffusion, and communication efficiency in a large-scale distributed satellite system. While the analysis is conducted within the proposed selective propagation

Table 1: Comparison of Decentralized Satellite Task Allocation Methods

Method	*BD [10]*	*NSS/DCOP [15]*	*m-ACCBA [6]*	*RF Selective Propagation (Proposed Method)*
Approach	Broadcast satisfaction and contention information	Decomposition and stochastic search	Auction-based iterative bidding	Reward-based selective propagation
Communication	Broadcast with a large amount of information	Scoped to local neighborhood	Exchanges across the network scale quadratically with the number of tasks	Broadcast with reduced information: reward functions, satellite identifiers, and timestamps
Computation	Light, simple heuristics only	Moderate, local search per agent	Heavy, evaluates all task permutations	Per task, light: reward prediction only
Scalability	Medium to high, bandwidth-limited	Higher than BD through standardized decomposition	Limited by bandwidth and task count, mitigated by m-ACCBA	Very high, demonstrated with thousands of satellites
Robustness	Medium, dependent on broadcast reliability	High, independent subproblems	High, supports asynchronous communication	High, opportunistic communication and dynamic topologies
Task and DSS knowledge at initialization	Tasks: global; satellite state: local heuristics only	Tasks: global; satellite state: neighbors only	Tasks: global; satellite state: unknown	Tasks: known to one satellite; satellite state: unknown

framework, the findings offer broader architectural insights: the results suggest that introducing even a small fraction of relay-capable nodes can significantly reduce convergence time and communication overhead, pointing to partial centralization as a promising direction for improving the responsiveness of distributed task allocation methods in general.

The main contributions are: (1) an extension of the single-task selective propagation framework [8] to a multi-task setting with multiple concurrent observation targets; (2) a parametric study of f_{CN} on convergence time, knowledge diffusion, and communication efficiency across constellations of up to 5,000 satellites and 500 concurrent tasks; and (3) the identification of a power-law relationship between η_L and n_{CN}/n_T, with coefficients scaling systematically with constellation size. Section 2 presents the methodology, Section 3 the experimental setup and results, and Section 4 the conclusions.

2 METHODOLOGY

This section describes the methodology used to develop the framework to investigate the effect of central nodes on multi-task allocation across multiple satellite architectures.

2.1 Problem Definition

We address the allocation of multiple tasks T, defined by observations of target objects O in SSO, among a satellite set S with different numbers of central nodes CN. Satellites operate in a Walker-Delta constellation, carry imaging payloads, and evaluate task execution capability and quality over the mission interval $\{t_{in}, ..., t_{end}\}$. The simulation assigns satellites in S to each task in T to maximize observation quality for satellites with good resource availability. At each timestep, the simulation evaluates feasible inter-satellite communications between satellite-central node pairs $(S, CN) \in S$. Initially, one satellite knows the full task set T. This assumption models two representative operational scenarios: tasks may be uplinked from a ground station to a single contact satellite, or originated autonomously by a satellite in orbit upon detection of target objects. Relaxing this assumption to account for multiple simultaneous task injection points, or for dynamic task generation during mission execution, is left as a direction for future work.

In both cases, knowledge must propagate through the network. Upon receiving a task, a satellite computes its RF value and broadcasts the higher of the received and evaluated values.

We define the task allocation problem as follows:

(1) $\{t_{in}, ..., t_{end}\}$ is the overall mission duration for the performance of the observations, starting at time t_{in} and ending at time t_{end}. t is discretized by a time step size $\Delta t > 0$ to result in the finite number of discrete time steps $t = (t_{end} - t_{in})/\Delta t$. The set of time steps, defined as $t = \{1, 2, ..., t_{end}\}$, contains all time steps at which a task can be performed.

(2) $S = \{1, 2, ..., n_s\}$ is a set of satellites, in a Walker-Delta Constellation architecture, equipped with an observational payload. The initial parameters and variables for the communication, power, and data-handling subsystems are identical across all satellites. From now on, any variable with the subscript i is associated with the i-th satellite.

(3) $f_{CN} = \{0.01, ..., 1\}$ is the fraction of the number of central nodes with respect to the total number of satellites in the constellation. They are responsible for the task propagation and are equipped with an observation payload.

(4) $O = \{1, 2, ..., n_O\}$ is the set of target objects to be observed, equally distributed along various SSO. Each object defines one observation task, so $n_T = n_O$.

(5) T_i, for each satellite $i \in S$, is the *Global Task Register*, storing for each task k the best known reward value $T_i^{k,RF}$, the associated satellite identifier $T_i^{k,ID}$, and the timestamp $T_i^{k,t}$ at which the task can be performed.

(6) L_i^k is the *Local Task Register* of satellite i, storing its locally evaluated reward $L_i^{k,RF}$ and identifier $L_i^{k,ID}$ for task k.

2.2 Architectural Variables and Parameters

The architectural design space was structured around one primary design variable, the fraction of central nodes, as defined in Eq. (1).

$$f_{CN} \triangleq \frac{n_{CN}}{n_s} \tag{1}$$

with $f_{CN} \in \{0.01, ..., 1.0\}$. A set of secondary variables supports the exploration of different architectures, including the number of satellites and target objects, as shown in Table 2.

Table 2: Architectural Design Variables

Tier	*Variable*	*Values*
Primary	Fraction of Central Nodes	0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0
Secondary	Number of Satellites	100, 500, 1000, 2000, 3000, 5000
Secondary	Number of Target Objects	1, 5, 10, 50, 100, 500

2.3 Figure of Merit

We consider five Figures of Merit (FoM):

- **Time to Converge** t_c represents the time to converge to a single satellite aware of being the best for each task.
- **Task Intensity** T_{int} defined as the ratio n_T/n_s and measures the load level of the network in terms of tasks to be performed.
- **Maximum Reward Function** RF_{Max} represents the highest performance achieved in performing a certain task.
- **Link Efficiency** η_L defined as the ratio between the total number of links n_L that occurred before convergence and the number of tasks n_T.
- **Diffusion Time** t_D represents the time elapsed until all satellites hold the complete task set T, i.e., $T_i \neq \emptyset$ for all $i \in S$. It marks the end of the knowledge propagation phase, which precedes and bounds the bidding convergence phase ($t_D \leq t_c$).

2.4 Task Packets

One assumption made at this stage of the research, to simplify simulation development, is that tasks are not executed individually in the framework but rather collectively as a *task packet*. The task-specific information exchanged, aside from communication

protocol overhead, consists of the Two-Line Element (TLE) of the target object, the satellite identifier associated with the highest reward function value, and the corresponding best-known reward value. Table 3 reports the assumed values for each component.

Table 3: Data Size Components

Variable	*Value [Bytes]*
TLE target object (s_{TLE})	104
ID + RF ($s_{ID} + s_{RF}$)	8
Transport Layer Protocol (s_{TLP})	34
Network Layer Protocol (s_{NLP})	24
Datalink Layer Protocol (s_{DLP})	35
Encoding Scheme Reed–Solomon (s_{ESRS})	13
TOTAL (D_s)	$112 \cdot n_T + 106$

As the number of tasks increases, the number of communication windows that are too short to send the entire set of tasks increases. To enable data exchange across a larger number of tasks, the communication protocol is assumed to follow principles similar to those of the Licklider Transmission Protocol (LTP) for deep-space communications. Satellite pairs initiate links and transmit data in minimum chunks corresponding to the amount transferable within a single timestep of duration $\Delta t = 1$ s. If a visibility window terminates before completion, satellites retain the already transmitted data and resume transmission during the next available contact.

2.5 Selective Propagation Algorithm

The selective propagation algorithm manages the information exchange and task allocation through pairs of satellites evaluated at each simulation timestep. A satellite pair (S_i, S_j) initiates communication only if at least one member is a central node, including pairs where both members are central nodes, neither satellite is processing previously received information, and the effective data rate of the inter-satellite link is strictly positive. Communication occurs only if one of the two satellites in the pair knows the tasks, or if there is a knowledge-mismatch condition, defined as either unequal task sets or disagreement on the best-known reward function value for at least one shared task. Data transfer accumulates across consecutive visibility windows until the full task set size is reached, at which point the final exchange is reached; partial transfers persist across interrupted links and resume upon reestablishment of contact.

Bidding Process. Upon receiving a task packet, the receiving satellite j enters a distributed single-item bidding process independently for each task k. The bid of satellite j for task k is its locally computed reward function value $RF_j^k(t)$, evaluated using Eq. (2) over the remaining mission interval $\{t, ..., t_{end}\}$. The satellite retains the task assignment; i.e., records itself as the best candidate, only if its locally computed bid strictly exceeds the best-known value received from the sender. Otherwise, the received entry is propagated unchanged. This mechanism enforces a greedy, communication-efficient consensus: at any point in time, each satellite stores the highest known bid for each task along with the identifier of the satellite that produced it.

During bidding, each satellite maintains a task buffer that temporarily stores newly received or locally updated task information within a timestep, isolating it from the global task register. The bidding logic separates incoming tasks into previously known and newly discovered subsets. For known tasks, larger reward values and their associated satellite identifiers overwrite buffered entries only when strictly greater values are received; ties are resolved by retaining the current buffered entry. For newly discovered tasks, the receiving satellite evaluates its local reward function over $\{t, ..., t_{end}\}$, then updates the buffered entry only if the locally computed value strictly exceeds the received one; ties are resolved in favor of the received entry, preserving the propagation direction. At the end of each timestep, buffered task information is promoted to the global task register for all satellites.

This mechanism enforces control over information diffusion, avoids redundant transmissions, limits communication to strictly necessary exchanges, and enables distributed convergence to a unique, best-performing satellite per task with the largest RF. Algorithm 1 summarizes the described selective propagation algorithm.

Reward Function: Evaluation of Observation Capability. Each satellite evaluates its capability for performing a given observation task k using a reward function RF, defined as in Eq. (2).

$$RF_i^k(t) \;=\; q_i\, a_i(t) \left[A\, p_i(t) \;+\; B\, ds_i(t) \;+\; C \left(1 - \left| \frac{\varphi_i^k(t)}{FoV_i} \right| \right) \right] \qquad (2)$$

For each satellite i, task k, and timestep t, the variables are defined as follows. $q_i \in \{0, 1\}$ is a binary indicator taking value 1 if the imaging payload is available and 0 otherwise; $a_i(t) \in \{0, 1\}$ takes value 1 if the satellite is not already committed to a concurrent task at timestep t and 0 otherwise. Both q_i and a_i act as multiplicative gates: if either is zero, the reward is zero regardless of the remaining terms. The remaining variables are continuous and defined over $[0, 1]$: $p_i(t) = E_{a,i,t}/E_{\max}$ is the fraction of stored energy available, where $E_{a,i,t}$ is the energy stored onboard satellite i at timestep t and $E_{\max}$ is the maximum storage capacity; $ds_i(t) = DS_{a,i,t}/DS_{\max}$ is the fraction of available onboard data storage, where $DS_{a,i,t}$ is the data storage available on satellite i at timestep t and $DS_{\max}$ is the total onboard storage capacity; and $(1 - |\varphi_i(t)/FoV_i|)$ is the normalized angular proximity of the target to the payload boresight, where $\varphi_i(t)$ is the angular separation between the pointing direction and the target position and FoV_i is the field of view of the imaging payload. The parameters A, B, and C are calibration weights that emphasize different aspects of the reward; in this study they are all set to 1. The maximum ideal value is $RF_{\text{ideal}} = 3.0$, attained when the payload is perfectly aligned with the target and the satellite has full energy, storage, and schedule availability. It shall be noted that this reward function definition has not significantly changed from previous work [8].

Convergence and Stopping Condition. The algorithm terminates when a global convergence condition is reached: for every task $k \in T$, all satellites in the network agree on the same satellite identifier as the best candidate, i.e., $T_{k,ID}^i = T_{k,ID}^j$ for all $i, j \in S$. This condition is checked at the end of each timestep after buffer promotion. When convergence is detected, the convergence time is

Algorithm 1 Selective Propagation Algorithm

Require: Satellites $S = \{1, \ldots, n_S\}$, relay-capable set $CN \subseteq S$, targets $O = \{1, \ldots, n_O\}$, connected to task set T, interval $[t_{in}, t_{end}]$, step Δt
Require: Coefficients A, B, C (default $A = B = C = 1$)
Ensure: Convergence time t_c; task register T_i for all $i \in S$
Note: $RF_i^k(t)$ is computed using Eq. (2) over $[t,\ t_{end}]$.
Init: Choose one satellite with T; all others start empty.
for $t \leftarrow t_{in}$ **to** t_{end} **step** Δt **do**
  Update states (positions, $p_i(t)$, $ds_i(t)$, $a_i(t)$ for all $i \in S$)
  for all pairs (i, j) with $i < j$ **do** ▷ i = sender, j = receiver
    if $i \in CN \ \vee \ j \in CN$ **then**
      if rate$(i, j) > 0$ **then**
        if knowledge mismatch between T_i and T_j, **or** $T_j = \emptyset$ **then**
          Satellite i sends task packet to satellite j;
            accumulate data until size D_s is reached
            (resume across interrupted links)
          if transfer complete **then**
            Satellite j enters processing mode (comm disabled for 1 s)
            for all received tasks k **do**
              if $k \in T_j$ **then** ▷ Task already known to j
                if $T_{k,RF}^{recv} > T_{k,RF}^{buf,j}$ **then**
                  $T_{k,RF}^{buf,j} \leftarrow T_{k,RF}^{recv}$
                  $T_{k,ID}^{buf,j} \leftarrow T_{k,ID}^{recv}$
                  $T_{k,t}^{buf,j} \leftarrow T_{k,t}^{recv}$
                end if ▷ Ties: keep current entry (no update)
              else ▷ Task newly discovered by j
                Compute $RF_j^k(t)$ using Eq. (2)
                $L_{k,RF}^{j} \leftarrow RF_j^k(t)$
                $L_{k,ID}^{j} \leftarrow j$
                if $L_{k,RF}^{j} > T_{k,RF}^{recv}$ **then**
                  $T_{k,RF}^{buf,j} \leftarrow L_{k,RF}^{j}$
                  $T_{k,ID}^{buf,j} \leftarrow j$
                  $T_{k,t}^{buf,j} \leftarrow t$
                else
                  $T_{k,RF}^{buf,j} \leftarrow T_{k,RF}^{recv}$
                  $T_{k,ID}^{buf,j} \leftarrow T_{k,ID}^{recv}$
                  $T_{k,t}^{buf,j} \leftarrow T_{k,t}^{recv}$
                end if ▷ Ties: received entry takes precedence
              end if
            end for
            Satellite j exits processing mode
          end if
        end if
      end if
    end if
  end for
  for all $i \in S$ **do**
    Promote buffer to T_i; clear buffer
  end for
  if converged (unique max $T_{k,RF}$ per k network-wide) **then**
    $t_c \leftarrow t - t_{in}$
    return t_c, $\{T_i\}_{i \in S}$
  end if
end for

recorded as $t_c = t - t_{in}$ and the simulation terminates. It should be noted that convergence of reward values alone is not sufficient: two satellites may store the same *RF* value for a task while associating it with different satellite identifiers, which would constitute a tie rather than a resolved allocation. The bidding process does not include a mechanism to discern between two identical RF values: an incoming reward value overwrites the buffered entry only if it is *strictly* greater. Consequently, an incoming RF equal to the currently stored value is discarded and the existing entry is retained unchanged, implying that the first satellite to propagate a given RF value for a task retains the assignment. This preserves the direction of propagation and ensures a unique identifier is stored network-wide upon convergence. Algorithm 1 summarizes the full procedure.

3 RESULTS

3.1 Experimental Setup

The simulation environment is configured as follows. Both satellites and targets are assumed to be in Low Earth Orbit. Table 4 summarizes the orbital parameters.

Table 4: Orbital Parameters for Satellites and Targets

Variable	*Satellites*	*Targets*
Type	Walker Delta	SSO
n_p [-]	10	n_T
a [km]	6923	7171
i [°]	53	98.55 ± 10
e [-]	0	0.0009
Ω [°]	$[0, 360]$	267
ω [°]	0	84
θ [°]	$[0, 360]$	$[0, 360]$

The constellation is composed of $n_p = 10$ orbital planes, among which the satellites are equally distributed. Each orbit is perfectly circular, with eccentricity $e = 0$, and has a semi-major axis $a = 6923$ km, corresponding to a constant orbital altitude of 550 km. The inclination i and the argument of periapsis ω are set to 53° and 0°, respectively, for all orbits. The initial true anomaly θ and the right ascension of the ascending node Ω take variable values. Given the total number of satellites n_S, satellite i is assigned Ω_i and θ_i according to Eq. (3) and Eq. (4).

$$\Omega_i = \frac{2\pi}{n_p} \left\lfloor \frac{i}{n_S/n_p} \right\rfloor, \tag{3}$$

$$\theta_i = \frac{2\pi}{n_S/n_p} \left(i \bmod \frac{n_S}{n_p} \right). \tag{4}$$

The outputs of (3) and (4) are converted from radians to degrees. The orbital parameters of the satellites correspond to the first group of Starlink satellites [13]. Each target defines a distinct orbital plane, with the reference plane corresponding to that of Envisat, characterized by an inclination of 98.55°. Target inclinations are equally spaced within a range of ±10° around this reference value. Initial target true anomalies are uniformly distributed over 360°. These represent orbits commonly used for Earth Observation satellites and those with the highest density of space debris [5]. The simulator precomputes all orbital states and retrieves them during execution. Each satellite carries identical subsystems. The electrical power subsystem includes a 0.4×0.3 m solar panel with 22% efficiency, packing factor 0.9, maximum stored energy $E_{\max} = 84$ kJ, operational power demand 28.5 W, and eclipse power demand 5.0 W, with variation across satellites depending on the orbit. The optical payload has a 5° field of view, aperture diameter $d_l = 9$ cm, and wavelength limits $\lambda = 400$–700 nm; assuming a target size $d_T = 10$ m, the Rayleigh criterion gives a maximum observable distance $d_{\max} = d_T d_l / (1.22\,\lambda) \approx 860$ km. The onboard computer

requires 1 s to evaluate reward functions, during which communication is disabled. Inter-satellite communications operate at 437 MHz with 9.6 kHz bandwidth, 2 W transmit power, and receiver sensitivity −151 dBW. Satellites establish communication only when the effective data rate, computed from the link budget, Shannon capacity, and bit error rate, is strictly positive; otherwise the link is inactive. All parameter values are selected to align with typical specifications for a 6U CubeSat. All satellites maintain a fixed attitude aligned with the positive z-axis of the Earth-Centered Inertial frame, with payload boresight constant over time. At each timestep of $\Delta t = 1$ s, the simulator updates the status of each satellite.

3.2 Comparison with Existing Methods

The results are interpreted in the context of Table 1. The proposed framework differs from the baseline methods along three dimensions directly observable in the results below.

Regarding *task knowledge at initialization*: BD [10], NSS/DCOP [15], and m-CBBA [6] assume global task knowledge from the outset. The proposed method relaxes this assumption entirely; only one satellite holds the full task set at initialization. The knowledge diffusion results in Section 3.3 quantify this cost: diffusion time t_D is the primary bottleneck, not the bidding process itself.

Regarding *scalability*: The proposed method evaluates only a single reward value per task per satellite, demonstrating convergence at up to 5,000 satellites and 500 concurrent tasks, a scale not demonstrated by any baseline method.

Regarding *centralization*: none of the baseline methods model or vary the degree of centralization as an architectural parameter. This study quantifies how f_{CN} affects convergence time, diffusion, and communication efficiency; a design dimension absent from the existing literature. The results show that $f_{CN} \geq 0.05$ is sufficient for fast convergence, suggesting partial centralization as a scalable architectural strategy applicable beyond the specific algorithm studied here.

3.3 Individual Run Analysis

We introduce the results of a selected individual run, for reference and to set a standard for the general performance of the framework. We select the constellation with 5,000 satellites, of which 10% (500) are Central Nodes, and with a set of 50 target objects.

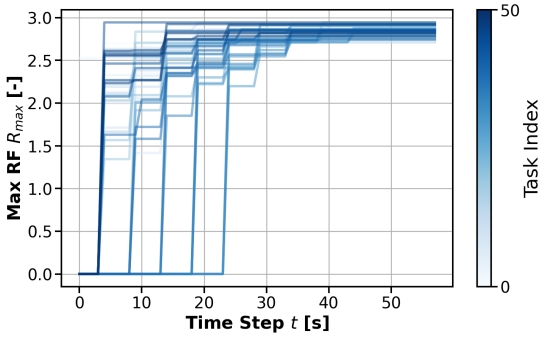


Figure 1: Reward function evolution over time for $n_S = 5000$, $n_T = 50$, $f_{CN} = 0.1$.

Figure 1 shows the maximum RF value per task, which reaches 75% of the theoretical maximum of 3.0 after 5 s; no further improvements are found after 45 s despite the network not yet having converged. Figure 2 shows knowledge diffusion, which completes at $t_D = 48$ s, only 10 s before full network convergence, indicating that task knowledge diffusion, not the bidding process, is the primary bottleneck under the current assumptions. Figure 3 shows the bidding status, which rises to a peak of over 40 simultaneous

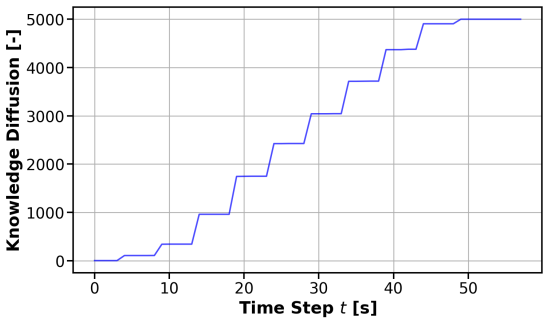


Figure 2: Knowledge diffusion over time for $n_S = 5000$, $n_T = 50$, $f_{CN} = 0.1$.

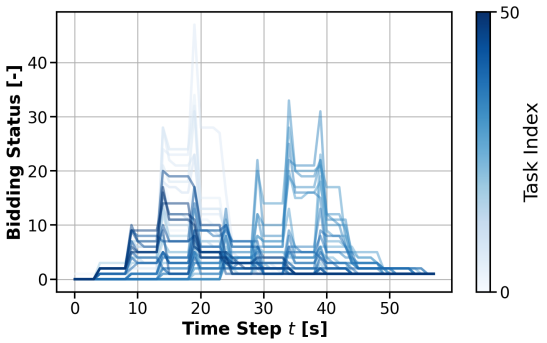


Figure 3: Bidding status over time for $n_S = 5000$, $n_T = 50$, $f_{CN} = 0.1$.

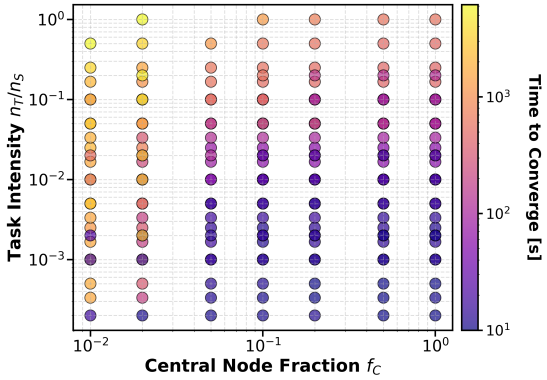


Figure 4: Performance Landscape of the studied collaborative framework, on log-log scale.

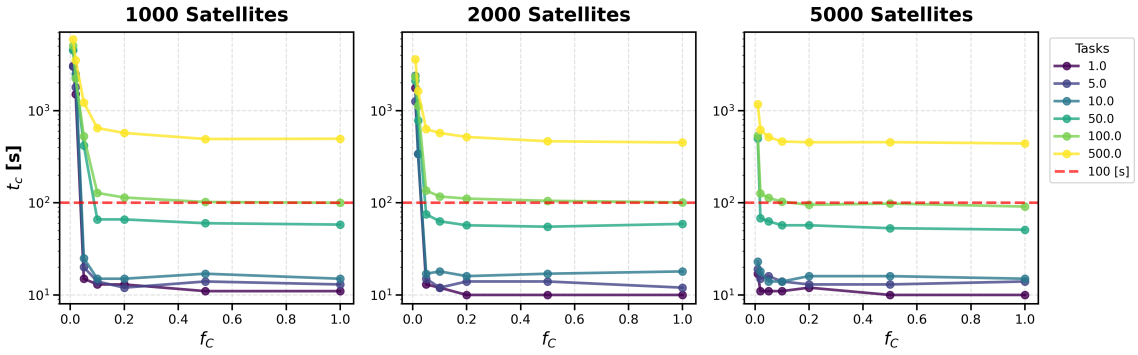


Figure 5: Time to converge t_c plotted against f_{CN} for $n_S \in \{1000, 2000, 5000\}$.

candidates per task during the initial propagation phase, then decreases nonlinearly until a single satellite per task is aware of being the best candidate.

3.4 Network Performance

The performance landscape plot in Figure 4 maps system behavior in a reduced two-parameter space. The horizontal axis represents the fraction of central nodes f_{CN}. The vertical axis represents task intensity, defined as the task load ratio n_T/n_S. Each data point encodes the convergence time t_c through color, with yellow indicating larger t_c values and purple indicating smaller t_c values. The surface shows a continuous dependence of network responsiveness on both f_{CN} and n_T/n_S. Increasing task intensity increases convergence time. Increasing f_{CN} decreases convergence time. The plot enables identification of the minimum f_{CN} to meet a target responsiveness level for a given task load. Configurations with $f_{CN} \geq 0.05$ maintain $t_c < 100$ s across the explored range of n_T/n_S.

Figure 5 reports convergence time t_c as a function of the central node fraction f_{CN} for multiple task counts n_T and three constellation sizes n_S. Each subplot corresponds to a fixed value of n_S.

The curves exhibit saturation behavior for $f_{CN} > 0.1$, where further increases in f_{CN} yield marginal reductions in t_c. Larger constellations show steeper reductions of t_c with increasing f_{CN}, indicating lower required central node fractions to achieve minimum convergence times as n_S increases. Increasing n_T reduces the variability of t_c with respect to f_{CN} for fixed n_S. The observed saturation suggests the existence of a minimum absolute number of central nodes required to achieve maximum network responsiveness for the given constellation geometry. A detailed analysis of the communication topology is outside the scope of this study.

Another performance metric considers the ratio of central nodes to tasks, n_{CN}/n_T. Figure 6 reports convergence time t_c (left panel) and link efficiency η_L (right panel) as functions of n_{CN}/n_T, with all axes in logarithmic scale.

The left panel shows that configurations with $n_{CN} < n_T$ do not achieve $t_c < 100$ s. This threshold appears invariant across the tested values of n_S, indicating a scale-independent constraint within the explored parameter range. The observed threshold can be understood as a large-scale network effect: as the number of tasks n_T increases, the task packet size D_s grows proportionally, requiring longer or more numerous communication windows to complete each transfer. With fewer central nodes than tasks ($n_{CN} < n_T$), the relay capacity of the network is insufficient to absorb this increased communication demand within the available contact opportunities, and the cumulative delay across the diffusion graph pushes convergence beyond 100 s. This effect is scale-independent: it persists across all tested values of n_S, suggesting that the ratio n_{CN}/n_T captures a fundamental constraint on the relay capacity of the network relative to its task load. Validation outside this range requires simulations with larger DSS.

Each horizontal stripe of points corresponds to a fixed number of tasks n_T, which compares with Figure 5. Movement along a stripe toward lower n_{CN}/n_T corresponds to decreasing n_{CN}. Increasing n_S reduces the sensitivity of t_c to variations in n_{CN} for fixed n_T. The color gradients converge toward linear trends as n_S increases, indicating reduced variability of convergence time with respect to central node count in larger constellations.

Communication Efficiency. The right panel of Figure 6 reports the link efficiency $\eta_L = n_L/n_T$, the ratio of total links before convergence to number of tasks. Lower η_L indicates higher information yield per communication event. As shown in Figure 6, η_L grows log-linearly with n_{CN}/n_T, with larger constellations exhibiting steeper slopes due to greater communication demand. This motivates the power-law fit:

$$\eta_L = A \times \left(\frac{n_{CN}}{n_T}\right)^b \tag{5}$$

Parameters A and b were identified for each constellation size n_S and plotted in Figure 7 as functions of n_S. The coefficient b follows a linear dependence on n_S (Eq. (6)), while A follows a power-law dependence on n_S (Eq. (7)). Linear regression yields p-values of 5.187×10^{-3} for b and 1.659×10^{-3} for A.

$$b = 4.15 \times 10^{-5}\, n_S + 0.617 \tag{6}$$

$$A \approx 1.1 \times n_S^{0.81} \tag{7}$$

Table 5 reports the total number of links n_L established before convergence under the current communication assumptions. The n_L increases for larger decentralized constellation. However, we see a maximum n_L at $n_T = 10$ and $f_{CN} = 1.0$, rather than at a boundary of the $n_T \times f_{CN}$ parameter space. This pattern holds across all tested

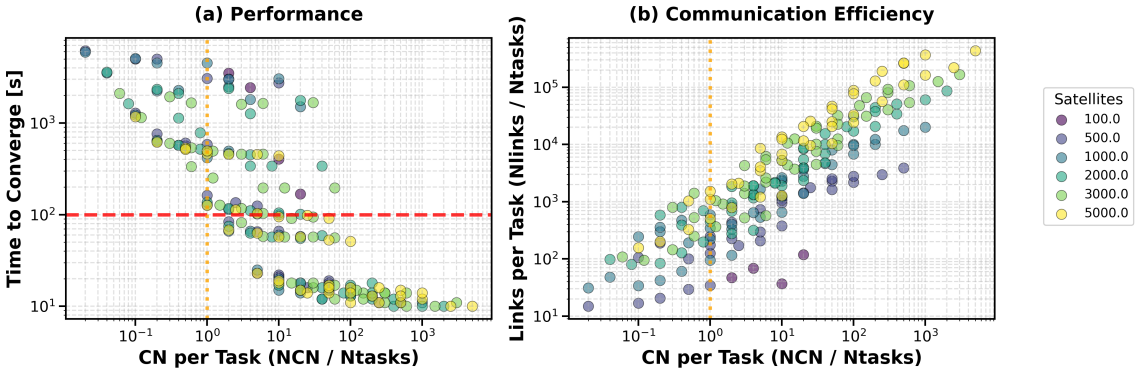
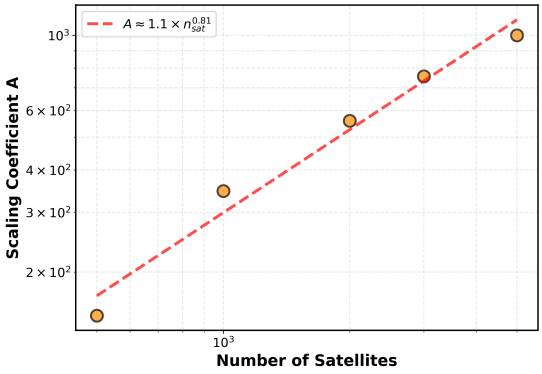
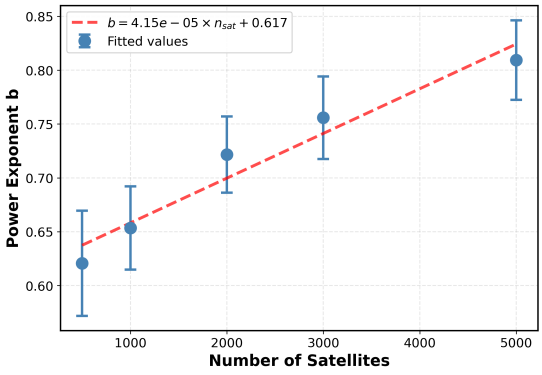


Figure 6: Time to converge t_c (on the left) and link efficiency η_L (on the right) against CN to task ratio. Axes are all in log-scale.



(a) Scaling coefficient A.



(b) Power exponent b.

Figure 7: Power-law coefficients A (top) and b (bottom) for the relation between η_L and n_{CN}/n_T as functions of n_S.

values of n_S and needs to be further investigated. Tables for other constellation sizes are omitted for brevity. For $n_T \leq 10$, the task packet size permits complete transmission within a single 1 s link, the minimum link duration imposed by the simulation timestep $\Delta t = 1$ s. Increasing n_T increases n_L, as satellites require additional communication rounds to reach agreement on multiple tasks. For $n_T > 10$, task packet sizes exceed the single-step transmission capacity. Satellites complete data exchange only after full task set transfer. Convergence time increases, link durations increase, and the number of completed links decreases. Parallel bidding across the network advances consensus during partial transmissions; some unfinished links are no longer needed yet still occur. This effect reduces n_L despite higher task counts.

Table 5: Number of Links for n_T and f_{CN} (5000 Satellites). Values are $\times 10^5$

f_{CN}	**0.01**	**0.02**	**0.05**	**0.10**	**0.20**	**0.50**	**1.00**
n_T							
1	0.25	0.33	0.56	1.1	1.6	2.2	4.4
5	0.62	1.0	2.1	3.9	6.5	1.3	1.8
10	0.86	1.4	2.6	4.7	8.9	19	26
50	0.76	1.0	2.0	3.4	5.8	10	12
100	1.0	1.1	2.1	3.5	5.2	9.6	10
500	0.79	1.0	1.6	2.5	4.0	6.0	6.4

The results show that $f_{CN} \geq 0.05$ maintains $t_c < 100$ s across all task intensities, and that n_{CN}/n_T is a scale-independent constraint: constellations with $n_{CN} < n_T$ consistently fail to converge within 100 s regardless of n_S. Larger constellations require smaller f_{CN} fractions for equivalent responsiveness, and communication efficiency follows a power-law in n_{CN}/n_T with coefficients scaling systematically with n_S.

4 FUTURE WORK AND CONCLUSION

Future work includes investigations into different constellation architectures, additional performance metrics such as system complexity and robustness, and determination of the optimal degree of centralization for specific use cases such as planetary boundary layer monitoring and wildfire detection. We will study resilience under random and targeted central-node failures. Extensions to dynamic task generation and multiple simultaneous task injection points will relax the single-origin assumption adopted here. We also plan to incorporate realistic inter-satellite network protocols to improve simulation fidelity. Overall, the results demonstrate that reward-based selective propagation enables fast, scalable, and communication-efficient multi-task allocation, highlighting the importance of partial centralization as a practical architectural strategy for future distributed satellite systems.

REFERENCES

[1] C. Araguz, E. Bou-Balust, and E. Alarcón. 2018. Applying autonomy to distributed satellite systems: Trends, challenges, and future prospects. *Systems Engineering* 21 (2018), 401–416. https://doi.org/10.1002/sys.21428

[2] M. K. Ben-Larbi, K. Flores Pozo, T. Haylok, M. Choi, B. Grzesik, A. Haas, D. Krupke, H. Konstanski, V. Schaus, S. P. Fekete, C. Schurig, and E. Stoll. 2021. Towards the automated operations of large distributed satellite systems. Part 1: Review and paradigm shifts. *Advances in Space Research* 67, 11 (2021), 3598–3619. https://doi.org/10.1016/j.asr.2020.08.009

[3] Alberto Candela, Jason Swope, and Steve A. Chien. 2023. Dynamic Targeting to Improve Earth Science Missions. *Journal of Aerospace Information Systems* 20, 11 (2023), 679–689. https://doi.org/10.2514/1.I011233

[4] Steve Chien, Itai Zilberstein, Alberto Candela, David Rijlaarsdam, Tom Hendrix, Aubrey Dunne, Aragon Oriol, and Miquel Juan Puig. 2025. Flight of Dynamic Targeting on the CogniSAT-6 Spacecraft. arXiv preprint. https://arxiv.org/abs/2509.05304

[5] Jian Huang, Xiangxu Lei, Bin Li, Jizhang Sang, and Hongkang Liu. 2022. Towards Fast and Reliable Evaluation of Detection Performance of Space Surveillance Sensors. *Remote Sensing* 14 (01 2022), 483. https://doi.org/10.3390/rs14030483

[6] Guoliang Li. 2020. Online scheduling of distributed Earth observation satellite system under rigid communication constraints. *Advances in Space Research* 65, 11 (2020), 2475–2496. https://doi.org/10.1016/j.asr.2020.02.018

[7] L. M. Marrero, J. C. Merlano-Duncan, J. Querol, S. Kumar, J. Krivochiza, S. K. Sharma, S. Chatzinotas, A. Camps, and B. Ottersten. 2022. Architectures and Synchronization Techniques for Distributed Satellite Systems: A Survey. *IEEE Access* 10 (2022), 45375–45409. https://doi.org/10.1109/ACCESS.2022.3169499

[8] Vincenzo Messina and Alessandro Golkar. 2025. Advancing Federated Satellite Systems Performance: A Collaborative Method for Improved Object Detection in Space. In *AIAA SCITECH 2025 Forum*. American Institute of Aeronautics and Astronautics, Orlando, Florida, USA. https://doi.org/10.2514/6.2025-0588 AIAA Paper 2025-0588.

[9] S. Oh and D. Vasisht. 2024. A Call for Decentralized Satellite Networks. In *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks*. Association for Computing Machinery, New York, NY, USA, 25–33. https://doi.org/10.1145/3696348.3696896

[10] S. Parjan and S. A. Chien. 2023. Decentralized Observation Allocation for a Large-Scale Constellation. *Journal of Aerospace Information Systems* 20, 8 (2023), 447–461. https://doi.org/10.2514/1.I011215

[11] J. A. Ruiz-De-azua, N. Garzaniti, A. Golkar, A. Calveras, and A. Camps. 2021. Towards federated satellite systems and internet of satellites: The federation deployment control protocol. *Remote Sensing* 13 (2021), 1–22. https://doi.org/10.3390/rs13050982

[12] Ryan S. Schaefer and Paul T. Grogan. 2024. Collaborative Constellation Analysis Framework for Wildfire Observing Missions. In *Proceedings of the IEEE Aerospace Conference*. IEEE, Big Sky, Montana, USA, 1–11. https://doi.org/10.1109/AERO58975.2024.10521362

[13] Calum Spring-Turner and Raj Thilak Rajan. 2022. Performance Bounds for Cooperative Localisation in the Starlink Network. arXiv:2207.04691 [eess.SY] https://arxiv.org/abs/2207.04691

[14] Josue I. Tapia and Paul T. Grogan. 2023. Dynamic Targeting for Precipitation Observing Missions: Integrating the GEOS-5 Nature Run Data Set. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2023)*. IEEE, Pasadena, California, USA, 3764–3767. https://doi.org/10.1109/IGARSS52108.2023.10282353

[15] Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. 2025. Decentralized, Decomposition-Based Observation Scheduling for a Large-Scale Satellite Constellation. *Journal of Artificial Intelligence Research* 82 (January 2025), 169–208. https://doi.org/10.1613/jair.1.16997

Dynamic Tool Generation for Autonomous Multi-Agent Systems in Space Missions

Dominik Opitz
German Aerospace Center (DLR)
Cologne, Germany
dominik.opitz@dlr.de

Carsten Hartmann
German Space Operations Center (DLR-GSOC)
Weßling, Germany
carsten.hartmann@dlr.de

Tobias Hecking
German Aerospace Center (DLR)
Cologne, Germany
tobias.hecking@dlr.de

Michael Felderer
German Aerospace Center (DLR)
Cologne, Germany
michael.felderer@dlr.de

ABSTRACT

Deep space missions are advancing increasingly further away from Earth, making continuous communication for mission maintenance progressively more difficult. To enable ongoing scientific exploration even without communication to Earth and to prevent mission abortion in critical situations, spacecrafts must be capable of operating with a high degree of autonomy. Achieving this level of autonomy requires intelligent systems that can reason about complex situations and interact effectively with their environment, often by leveraging specialized tools.

Current autonomous agents, however, typically rely on predefined tools to interact with their environment, which limits their flexibility in novel or unexpected situations. In this work, we present a framework that enables LLM-based agents to dynamically generate Python tools at runtime, allowing autonomous reasoning and task execution in previously unseen environments - a crucial capability for deep space missions. Our system connects a dedicated Tool-Agent with a planning agent to form a fully autonomous pipeline capable of planning and executing unseen tasks without predefined tools or task-specific guidelines. The system functions with standard mid-sized LLMs (up to 30B parameters), without pre-training or in-context learning. We demonstrate this approach on a space debris scenario, where a satellite detects nearby debris; our system successfully assesses collision risk by autonomously creating, executing, and analyzing tools for trajectory calculation of the two objects.

While this work is still a proof-of-concept, it highlights the potential of runtime tool generation to enhance the autonomy of agents operating in deep space, enabling more resilient and adaptive mission operations.

KEYWORDS

LLM, Tool-Generation, Multi-Agent Systems, Space Systems

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

ACM Reference Format:
Dominik Opitz, Carsten Hartmann, Tobias Hecking, and Michael Felderer. 2026. Dynamic Tool Generation for Autonomous Multi-Agent Systems in Space Missions. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 10 pages.

1 INTRODUCTION

We propose a framework that integrates a Planning-Agent with a dedicated Tool-Agent capable of dynamically generating Python scripts at runtime. The Planning-Agent maintains a high-level task plan that is continuously updated with gained knowledge and subtasks as needed. The Tool-Agent focuses on executing delegated subtasks by creating and utilizing tools when required. It is tightly coupled to the Planning-Agent and can request additional input from it at any time during task execution. Tools are stored and reusable so that they can be used multiple times and in sequence. Complex and large outputs from tools are stored in dedicated artifacts, keeping the agents working context manageable. Both agents condense their interaction history after logical steps, preserving task-relevant information and supporting focused reasoning like that of episodic memory. In contrast to related works, our framework does not rely on the use of external services and heavy processing in the cloud. Instead it employs small-sized agents up to 30B parameters that can be operated on platforms with constraints on computational resources, memory, and energy consumption, which makes it suitable for increasing autonomy of remote space systems operating under uncertain conditions.

In summary, our contribution is as follows:

- A multi-agent architecture combining high-level planning with dynamic, on-demand tool generation,
- A mechanism for safe and reusable tool execution with artifact-based state management,
- An approach for maintaining concise, task-relevant memory to support autonomous reasoning in complex, novel environments.

We demonstrate our system with a scenario, where the system is used to assess the collision risk of space debris with a satellite. In general, the field of spacecraft collision avoidance studies the process of minimizing or mitigating the risk of collisions between

spacecraft in Earth orbit and other orbiting objects. These orbiting objects can be other spacecraft, remnants of satellite collisions, or naturally occurring materials, like micrometeoroids and natural satellites. The European Space Agency (ESA) currently tracks around 15,860 (man-made) satellites in Earth orbit, of which about 12,900 are still functional. Furthermore, the overall number of objects in orbit is 54,000 space objects greater than 10 cm, 1.2 million objects between 10 cm and 1 cm, and 130 million objects between 1 mm and 1 cm [12]. While objects of very small size seem noncritical, a potential impact with another object can have catastrophic results, due to the high relative velocity and thus high kinetic energy. For example, the orbital velocity in low Earth orbit (LEO) is around $7.8km/s$. Therefore, two objects perpendicular to each other would collide at roughly $12.2km/s$. Even tiny particles like paint flecks or solidified liquids expelled from spacecraft could permanently damage, or, in the worst case, disintegrate another object. As such, these collisions should be avoided at all cost. However, this currently requires tremendous effort, from building and maintaining ground infrastructure for the tracking of objects, to the computational effort of collision prediction methods, to the organizational and operational effort to coordinate, plan and execute an avoidance maneuver. This is further complicated if one of the objects cannot be remote controlled (e.g., due to being defunct), or there isn't sufficient time to act. While these circumstances already justify the need for increased autonomy, the problem of collision avoidance is predicted to become even worse, with many public companies and national institutions proposing or launching mega-constellations of satellites, like SpaceX's Starlink, Amanzon's Leo (formerly known as Project Kuiper), or China's Guowang satellites. ESA suggests, that, even without additional launches, catastrophic collision numbers will increase, due to the fact that collision events add debris objects faster than debris can naturally re-enter the atmosphere [12].

To deal with this increase in collision events, future spacecraft might be equipped with autonomous on-board capabilities to detect other objects, predict the point and time of closest approach (PCA and TCA) and perform avoidance maneuvers to prevent potential collisions. More specifically, the PCA is defined as the point in each object's orbit, where the magnitude of the relative position vector between the two objects is a minimum, and the TCA is defined as the time, at which the minimum miss distance between the two objects occurs. In the system that is developed as part of this paper, specifically the task of collision prediction is demonstrated, by utilizing dynamic tool generation. For the purposes of this, we refer to the controlled asset (i.e. the spacecraft performing collision avoidance) as the primary object and the object it encounters as the secondary object. Furthermore, encounters can be classified in two different scenarios: 1. short-term and 2. long-term [18]. On the one hand, short-term encounters usually happen between objects with two significantly different orbits, which results in very high velocities at the point of closest approach and lasting a few seconds only. On the other hand, long-term encounters happen between two satellites traveling along near identical orbits. For the purposes of demonstrating our system only the first scenario of encounter is considered. The goal is to combine a tool-agent with a planning agent, in order to dynamically generate Python code, that can solve the task of finding the PCA and TCA, without providing predefined tools.

2 RELATED WORK

Since the introduction of Large Language Models (LLMs), we are witnessing a progressive shift away from passive chatbots with static knowledge towards active agents with interactive capabilities. A significant part of research focuses on *Agents* that leverage predefined tools to extend their reasoning and interaction abilities. Such tools enable them to perform tasks such as query databases, retrieve web information or perform structured calculations. Since such approaches are limited by the tools that are provided to them, the agents are constrained in their flexibility when faced with novel and unexpected tasks or data. For this reason, recent work has begun exploring dynamic tool generation, a concept where tools are no longer required before runtime, but instead dynamically synthesized during the reasoning process.

Among the earliest adopters in the realm of tool-generating systems is *ATLASS* [7]. *ATLASS* is a closed-loop framework designed for problem solving tasks with tool learning and selection. The system is divided into three stages to (i) determine whether tools are required, (ii) generate or retrieve required tools and (iii) execute the respective tools in order to complete the initial task. The Tool Generation process is designed as an iterative loop that involves the installation of dependencies and writing of Python code until the code execution runs without errors. *ATLASS* is designed with the GPT-4.0 model from OpenAI, a large, cloud-based model.

Wölflein et al. [17] present *ToolMaker*, a framework that turns scientific papers and their accompanied code repositories into functional, LLM-compatible tools, facilitating reproducibility of scientific publications. Their system follows an iterative workflow inside a dedicated execution environment to handle repository installation, dependency management and function generation. Wölflein et al. demonstrate how agents can go beyond the generation of simple tools and instead "produce tools for real-world scientific tasks". *ToolMaker* was designed around and evaluated on several large-scale cloud models, namely GPT-4o, GPT-o3-mini and Claude 3.5 Sonnet.

While *ATLASS* and *ToolMaker* use prompt techniques to enable the generation of tools, *DeepAgent* [10] introduced by Li et al. focuses on discovering and invoking existing tools from massive tool sets. Rather than creating new tools, *ToolMaker* uses a retrieval system to match relevant tools and their documentation for a particular task. In order to improve tool-use in subsequent requests, the internal reasoning models maintain a tool memory to store executed tools along with metadata such as success rates and parameter combinations. In their study, Li et al. used the QwQ-32B model [15] as their main reasoning LLM.

Recent work has further introduced a paradigm shift away from purpose-specific tools towards the acquisition of broader skill-sets. *CASCADE* [8] introduces an agentic system that learns to establish executable scientific routines that are autonomously developed and stored for future use. Here, an essential concept is the agents ability to acquire skills through pre-defined "meta-tools" such as web search and code execution. Evaluated on several of OpenAI's

models, Claude and a Qwen 30B model, *CASCADE* significantly outperforms traditional "tool-use" baselines.

The aforementioned frameworks primarily target terrestrial research or labratory environments. Systems like ATLASS and ToolMaker rely on high-resource cloud-based models, a requirement that is misaligned with the constraints of space systems, which operate under severe computational limitations. Instead, our framework is specifically optimized to function with standard mid-sized LLMs (up to 30B parameters) that can be operated on modern edge-processors. In contrast to CASCADE and DeepAgent, which depend on persistent external connectivity for web-based learning or large-scale API retrieval, our architecture is designed to operate independently from external resources. Rather than discovering existing tools through pre-computed indices, as in DeepAgent, or learning and reusing complex external routines, as in CASCADE, our system employs a dual-agent pipeline that dynamically generates Python scripts at runtime. These scripts directly process raw environmental data without relying on predefined task-specific instructions. Thus, our approach can be a building block of autonomous systems operating on spacecraft where energy consumption is a crucial factor as well as in in isolated environments, such as deep space missions where persistent high bandwidth communication with Earth is limited.

3 MOTIVATING EXAMPLE

To illustrate the practical utility of our framework, we employ a scenario where a satellite (primary object) detects a nearby piece of space debris (secondary object) and must independently assess the risk of collision. As mentioned in the introduction (ref. ch. 1), this is referred to as a short-term encounter, and the goal of the system is to find the point of closest approach. Provided with only two location measurements for each object, the system receives the following task:

> *A new piece of space debris has been detected in the vicinity of our satellite orbiting Earth. For the satellite and the debris, two location measurements have been recorded at two different timestamps, stored in the 'satellite_measurements.csv' and 'debris_measurements.csv' files within the 'trajectories' directory.*
> *Determine whether the newly detected space debris poses a collision threat to the satellite.*

For context, the tracked location data are summarized in Table 1. Measurements for both the satellite and the debris include the columns `timestamp`, `x`, `y`, and `z`. The debris data additionally contain a `mass` column, which is irrelevant to the task but introduces minor noise that the agents should ideally ignore.

Our multi-agent setup successfully identifies both the time and point of closest approach (TCA and PCA), yielding a minimum distance of 21.38 units after 11.43 units of time. A visual, step-by-step trace of the agent interactions is provided in Appendix A. Unlike approaches that rely on in-context LLM calculations - which are often unsafe or approximate - our agents outsource all critical computations to Python tools, ensuring precise and reliable results. Notably, the agents autonomously determined that they needed to access and extract measurement data from the provided CSV files, a task that LLMs cannot typically perform on their own. Importantly, these tools were not predefined or externally instructed; the agents themselves identified the necessity of each tool and generated or executed it as required, demonstrating the key advantage of our dynamic, self-directed tool-generation approach.

timestamp	x	y	z	mass
Debris measurements				
0	40	-40	-5	20
10	40	0	15	20
Satellite measurements				
0	0	0	-5	—
10	20	0	5	—

Table 1: Location measurements for debris and satellite objects.

At a high level, the agents proceed as follows (see also Appendix A):

1. The Reasoning Agent initializes a high-level plan consisting of four steps: (1) load and interpret measurements, (2) calculate trajectories, (3) predict closest approach, and (4) assess collision risk.
2. The Reasoning Agent delegates the first execution task to the Tool Agent, requesting the extraction of the measurements of the satellite and debris locations from file.
3. The Tool Agent fulfills this request through the following substeps:

3.1. It generates the tool `inspect_csv_files(file_paths)` to inspect the structure of the input data. The tool determines that the satellite measurements contain the columns `timestamp`, `x`, `y`, and `z`, while the debris measurements additionally include a `mass` column.

3.2. Based on the observed structure, the Tool Agent generates a second tool, `extract_trajectory_data(file_paths)`, which extracts the relevant position measurements. The `mass` column is correctly ignored, as it is not required for the task. The extracted measurements are stored in artifact `42b`.

4. Following the plan, the Reasoning Agent delegates the next task to the Tool Agent, instructing it to compute the trajectories of both objects.
5. To do this, the Tool Agent first generates a new tool called `calculate_trajectory(measurements)`, which computes position and velocity vectors as well as final positions for each object. The tool is then executed using the measurement data from artifact `42b`, and the resulting trajectory parameters are stored in artifact `6f8`.
6. The Reasoning Agent then delegates the computation of TCA and PCA to the Tool Agent.
7. To complete this task, the Tool Agent generates the tool `calculate_closest_approach(trajectory_params)` and executes it using the trajectory parameters stored in artifact `6f8`. The initial execution fails due to a key error; the tool is subsequently repaired and re-executed successfully. The correct TCA and PCA values are stored in artifact `475` and returned to the Reasoning Agent.

8. With all planned steps completed, the Reasoning Agent finalizes the assessment and terminates the process.

4 METHODOLOGY

We illustrate the setup of our proposed framework in Figure 1. The frameworks' core is composed of the *Reasoning Agent* and the *Tool Agent*. The Reasoning Agent is responsible for the maintenance of the overarching goal, task decomposition and task delegation. The Tool Agent independently executes dedicated, more isolated tasks directly delegated by the Reasoning Agent. It also coordinates the generation and execution of tools, working together closely with the Coding Agent used for the generation of tools (Python scripts).

Once created, tools are stored alongside augmented information describing the tools purpose and usage. Executed tools produce persistent artifacts that can be viewed and invoked directly within tool calls. Finally, our system employs an *Epoisodic Summarizer* to condense the reasoning history of the Reasoning Agent and Tool Agent after logical steps. This approach has proven successful in maintaining focused task execution [10, 11, 16].

Generally, we aim to utilize models as small as possible as practical deployment in space environments is constrained by limited onboard computational resources.

4.1 Components

In this subsection, we introduce the main components of our framework. For this purpose, we focus on outlining their role within the overall architecture.

Reasoning Agent The Reasoning Agent is responsible for planning and coordinating the steps required to reach a defined goal. It maintains a continuous plan to organize sequences of tasks, delegate them to the Tool Agent and store facts and knowledge gathered through the process. As this role requires more general and broader commonsense reasoning abilities, we base the Reasoning Model on the `Gemma 27B` LLM [13].

Tool Agent Within our framework, the Tool Agent operates as an execution-oriented component that fulfills subtasks delegated by the Reasoning Agent. From the perspective of the Reasoning Agent, it is treated as a black box: The Reasoning Agent specifies **what** information is required or what needs to be done, but remains agnostic in **how** that information is obtained or a task is executed. The Tool Agent's novel capability is the autonomous coordination, generation and execution of tools. It uses four specialized commands to initiate any of these actions, which is detailed in Section 4.2. The agent does not generate the raw source code of tools itself, but rather manages the coordination and use around tools. As such, it is assigned with a `Qwen 30B` LLM [14], a model which we found delivers straightforward reasoning steps, ideal for this use case.

Coding Agent The Coding Agent is directly responsible for (i) generating the source code of new tools, (ii) classifying the reason for failed tool executions and (iii) repairing tools that have failed after execution, if the failure reason was assessed to be an issue in the code. Due to its highly focused responsibilities, the Coding Agent is based on a `Qwen-Coder 14B` [9] model by default. In the case of repeated generation of invalid code, the model dynamically increases to its larger `Qwen-Coder 30B` variant.

Tools Tools are isolated Python scripts intended to support specific, encapsulated use cases, ranging from basic operations such as local file access to advanced tasks involving mathematical computation and external data retrieval. Each tool execution yields artifacts containing (i) the raw output, (ii) associated meta-information (e.g., data type), and (iii) a verbalized summary when the output cannot be passed directly to the Tool Agent, such as in cases of large outputs or non-serializable data objects (e.g., database clients or library instances). Tools are invoked by the Tool Agent with a defined set of parameters, which may be specified directly or populated via references to artifacts produced by prior tool executions. This design supports sequential tool execution and the compositional integration of tool outputs.

Episodic Summarizer Finally, an episodic Summarizer LLM continuously condenses the evolving conversational context shared between the Reasoning Agent and the Tool Agent. This strategy has been shown to effectively improve the agents' reasoning ability in several prior studies [10, 11, 16]. During the agents' reasoning process, the lightweight language model LLaMA 8B [1] incrementally organizes the accumulated reasoning history into discrete logical steps. Such steps may include, for instance, tool generation, tool execution, or intermediate planning phases. Each step typically comprises multiple prompts and intermediate reasoning traces, which are consolidated into a single concise summary once the step is completed.

4.2 Workflows

In this subsection, we focus on the core principles of our workflow. Specifically, we detail the process of task planning, tool generation, tool execution and artifact handling.

Task Planning and Delegation Throughout the process, the Reasoning Agent maintains a dynamic plan aimed at fulfilling its overarching goal. The plan contains the overarching goal, the relevant substeps required to complete the overarching goal as well as a dynamic memory used to store information, facts and other data gathered throughout the process. The Reasoning Agent can update the plan at any time during the process, whenever it deems this as necessary. To facilitate the update of the plan and delegation of individual tasks, the Reasoning Agent can use the commands `<UPDATE_PLAN>` and `<DELEGATE>`, which will trigger individual prompt sequences to fulfill this request. As of now, tasks can only be delegated to the Tool Agent, as no other agents are available. However, the principle can be extended to using several additional agents. This would allow the delegation of tasks to most appropriate agents and enable parallel work.

Tool Generation The process of Tool Generation is initiated by the Tool Agent whenever it decides that a new tool is required to fulfill a specific task. The agent will trigger the process by streaming a special `<CREATE_TOOL>` command. The agent will then be asked to provide a tool specification outlining name, purpose, input parameters and expected output of the tool. The specification template is illustrated in (Listing 1). It is forwarded to the Coding Agent, which will generate the source code of the tool based on the specification.

Tools are validated based on five metrics:

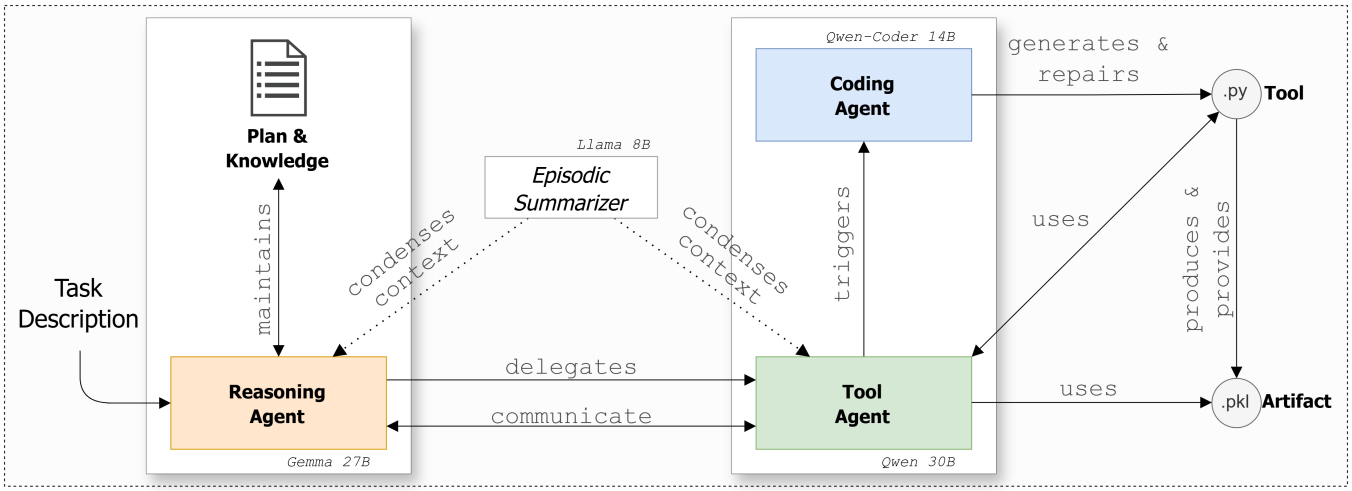


Figure 1: Overview of our framework architecture. The system consists of two main agents: Reasoning Agent and the Tool Agent. A Coding Agent generates tools as directed by the Tool Agent. Executed tools produce persistent artifacts for later use. An episodic summarizer LLM condenses each agent's reasoning history after logical steps.

```
{
     "name" : "<tool name>",
     "description" : "<tool description>",
     "parameters" : {
        "<parameter name>": {
             "type" : "<datatype>",
              "description" : "<param. description>",
              "constraints" : "<constraints>"
        }
    },
     "expectedOutput" : "<expected output>"
}
```

Listing 1: Tool specification format

(1) **Syntax Validation:** Ensures that the generated tool constitutes syntactically valid Python code by attempting compilation without execution.
(2) **Entry-Point Validation:** Verifies the presence of a designated `main()` function, which serves as the required entry point for tool execution.
(3) **Import Validation:** Analyzes all import statements to detect potentially unsafe modules. Additionally, this step ensures that all imported dependencies are available in the execution environment.
(4) **Executability Validation:** Checks whether the tool can be executed without raising runtime errors by running the code in an isolated namespace; for example, tools that trigger exceptions such as division-by-zero or unresolved attribute accesses during execution are rejected.
(5) **Call-Signature Validation:** Validates function calls to imported modules by resolving their call paths and verifying that the provided arguments conform to the corresponding function signatures; for instance, calls supplying an incorrect number of arguments or invalid keyword parameters are flagged.

Once the tool is validated, it is added to a central registry. Here it remains available and the Tool Agent can use it in subsequent steps as many times as needed. Listing 2 (`inspect_csv_files`) and Listing 3 (`calculate_closest_approach`) illustrate two examples of tools that were generated within the process of our Motivating Example (Section 3). Specifically, `inspect_csv_files` allows the agent to inspect the structure of the .csv file containing the location measurements, which is a required step to understand how the data can be processed. The tool `calculate_closest_approach` allows the agent to calculate the trajectories of the two objects based on the previously calculated position and velocity vectors. In our use case, the tools were created by the `Qwen-Coder 14B` LLM of the Coding Agent.

```
import csv
def main(file_paths):
    result = {}
    for file_path in file_paths:
        with open(file_path, mode='r', newline='',
     encoding='utf-8') as csvfile:
            reader = csv.reader(csvfile)
            headers = next(reader)
            result[file_path] = headers
    return result
```

Listing 2: Example of a Generated Tool: inspect_csv_files

```
import numpy as np
def main(trajectory_params):
    # Extract trajectory parameters
    # The keys in trajectory_params are the file paths,
     so we need to identify which is satellite and which
     is debris
    # For this implementation, we'll assume the first two
      keys are satellite and debris respectively
    keys = list(trajectory_params.keys())
```

```
    satellite_position = np.array(trajectory_params[keys
     [0]]['position'])
    satellite_velocity = np.array(trajectory_params[keys
     [0]]['velocity'])

    debris_position = np.array(trajectory_params[keys
     [1]]['position'])
    debris_velocity = np.array(trajectory_params[keys
     [1]]['velocity'])

    # Calculate relative position and velocity
    relative_position = satellite_position -
     debris_position
    relative_velocity = satellite_velocity -
     debris_velocity

    # Calculate the time of closest approach using the
     formula:
    # t = -(r0 . v) / (v . v)
    r0_dot_v = np.dot(relative_position,
     relative_velocity)
    v_dot_v = np.dot(relative_velocity, relative_velocity
     )

    if v_dot_v == 0:
        raise ValueError("Relative velocity is zero,
     closest approach cannot be determined.")

    t_closest_approach = -r0_dot_v / v_dot_v

    # Calculate the position at the time of closest
     approach
    satellite_position_ca = satellite_position +
     satellite_velocity * t_closest_approach
    debris_position_ca = debris_position +
     debris_velocity * t_closest_approach

    # Calculate the distance at the time of closest
     approach
    distance_ca = np.linalg.norm(satellite_position_ca -
     debris_position_ca)

    # Return the result as a dictionary
    return {
        'closest_approach_time': t_closest_approach,
        'minimum_distance': distance_ca
    }
```

Listing 3: Example of a Generated Tool: calculate_closest_approach

Tool Execution Similar to tool generation, tool execution is initiated by the Tool Agent via a dedicated <EXECUTE_TOOL> command. This command triggers the tool invocation process, whose detailed process is illustrated in Figure 2. Broadly, it can be decomposed into three logical stages:

(1) **Parameter Specification:** The Tool Agent first identifies the tool to be executed and provides a brief reason for the invocation, which is later used to generate a summary of the output. The agent then supplies the required input parameters, which may either be specified explicitly (e.g., strings or numerical values) or derived from the outputs of previously executed tools by referencing their associated artifacts.

(2) **Tool Execution and Output Handling:** The selected tool is executed with the provided parameters. Each tool is required to expose a single `main()` function as its sole entry point, simplifying and standardizing the execution procedure. If the execution was successful, the output is directly stored as an artifact containing the raw result, data type, handle, size, and - if the raw output is excessively large or cannot be serialized (e.g., complex objects such as database clients) - a concise verbal description generated by a lightweight language model (LLaMA 8B [1]).

(3) **Error Handling and Recovery:** If tool execution fails, the Coding Agent analyzes the failure to determine whether it is likely caused by errors in the tool's source code or by invalid input parameters. Based on this assessment, the agent proposes either a code-level repair or a revision of the input parameters. In cases where the source code can be repaired, the corrected version is executed automatically; otherwise, responsibility for adjusting the input parameters is delegated back to the Tool Agent.

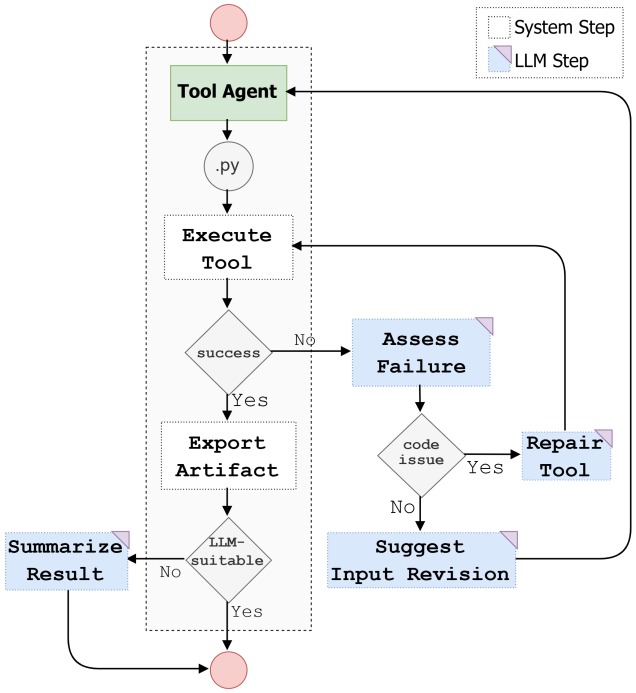


Figure 2: Process of Tool Execution

Artifact Handling By enabling the creation of virtually arbitrary tools, tool outputs may assume unanticipated forms and data types. In general, large language models (LLMs) are limited to processing serializable, text-based inputs; when working with predefined tools, this constraint can typically be enforced by design.

However, once agents are empowered to generate tools dynamically, the resulting output types become inherently unpredictable and are often incompatible with direct consumption by an LLM. For example, a database client or figure object are incompatible with LLMs, as they are not text-based. While larger models may exhibit an implicit awareness of these limitations and could consequently construct tools to output only compatible outputs, smaller language models lack the ability to anticipate whether a tool's output will

```
"artifact_id": "artifact_475",
"summary": "Tool result of tool
    calculate_closest_approach",
"context": "Result stored as artifact (dict)",
"content": {
    "closest_approach_time": 11.4285,
    "minimum_distance": 21.3808
},
"handle": "/artifacts/artifact_475.pkl"
```

Listing 4: Artifact Representation Created from Tool Output (Decimal values rounded to 4 decimal places for readability)

be produced in a consumable format. To address this challenge, our system introduces an explicit artifact abstraction that mediates between tool execution and agent reasoning. Tool outputs are encapsulated into serializable artifacts with a semi-fixed structure, consisting of the raw tool result, its data type and size, a handle for later reference, and a concise verbal description generated by a lightweight language model. Whenever the tool output itself is serializable, it is forwarded directly to the Tool Agent; only in cases where the output cannot be serialized do we fall back to the verbalized description, which provides a high-level semantic summary without exposing an incompatible internal representation. Listing 4 showcases the representation of artifact 475, representing the output of the tool `calculate_closest_approach` (Listing 3). The raw output (`content`) is a small dictionary containing the TCA and PCA of the two objects. As this output format is directly consumable by the Tool Agent, no additional textual description is required here. For LLM-incompatible outputs, the constructed verbal description cannot describe the raw content of the output. Instead, it can describe its metadata and purpose. For example, in case of a database client, the description might state the type of client and how it can be used in other tools.

Overall, our approach departs from the conventional paradigm in which agents directly operate on tool outputs. Instead, we adopt an artifact-centric interaction model in which agents reason over structured representations of results rather than raw outputs themselves. This shift decouples agent reasoning from the concrete data representations produced by tools, enabling robust handling of heterogeneous and non-serializable outputs while preserving composability across sequential tool invocations.

5 DISCUSSION

5.1 Advantages of Dynamic Tool Generation

The core contribution of this work is a dynamic tool generation process, autonomously triggered by LLM-based agents. The primary value of this framework is not necessarily to produce code thatis more computationally efficient than human-written code, but to produce code that is adapted to unseen data schemas and scenarios. In the motivating example, the generated tool automatically ignored the irrelevant 'mass' column in the CSV file. A pre-defined human tool might have requird explicit error handling for unexpected columns. Therefore, the distinction lies in rovustness to novelty rather than raw performance. This approach allows agents to overcome two key limitations of tool-based LLMs.

First, the LLMs remain restricted to the tools that were provided to them. Hence, they cannot dynamically adapt them to new or evolving situations. It is infeasible to anticipate every possible scenario in advance and create mechanisms for each one that the LLM can use. On one hand, even minor deviations from nominal use cases - such as adapting a maneuver-planning tool to different propulsion models - would require explicitly predefined tools for each variation. On the other hand, entirely new problem classes may emerge, such as the sudden need to analyze an unprecedented orbital regime or a previously unconsidered debris interaction scenario.

Second, when an LLM faces a situation requiring more advanced (e.g. mathematical) reasoning, it is forced to perform those calculations within its own internal framework, as long as no tool exists for that specific purpose. While modern LLMs have made significant advancements in reasoning, these types of calculations are inherently prone to error. Unlike specialized tools that perform precise computations, LLMs do not actually "calculate" but instead "reason" through them, leading to potential hallucinations or inaccuracies. Figure 3 illustrates this imprecise nature of LLMs when presented with calculations. We prompted the LLMs individually with the task of calculating the TCA and PCA based on the same provided measurements. Their calculations are often approximate, but not precise. This example illustrates one of the major benefits of tool-generating LLMs: it enables them to offload critical computations to external, reliable systems, thus ensuring more accurate results and reducing the cognitive load on the LLM itself.

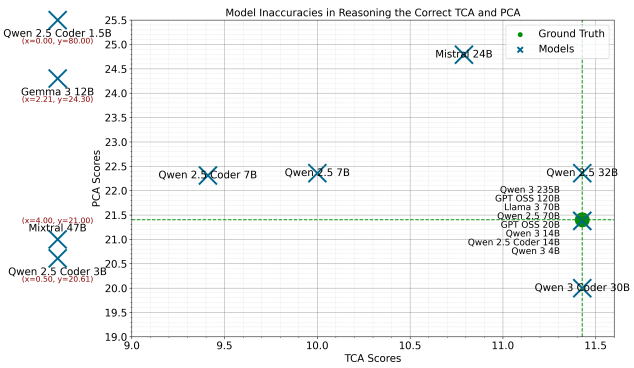


Figure 3: Model Inaccuracies in Reasoning the Correct TCA and PCA. We prompted all models with the same, neutral prompt instructing it to identify the TCA and PCA based on the measurements. The scatter plot displays their calculated TCA and PCA.

5.2 Limitations & Challenges

While the benefits of dynamic tool generation can fundamentally advance the capabilities of agentic systems, our research uncovers new challenges that have to be addressed.

Tool Output Handling. Among the most prominent hurdles of our system is the handling of tool outputs. For conventional approaches, tool outputs are well structured and can be easily formatted into a form that LLMs can interpret - typically serializable, formatted text. In our framework we encountered the challenge

of agents generating tools that produce outputs which are inherently incompatible with the LLMs text-based input requirements. Examples of such cases include tools that output database clients, pandas dataframes or image files. Further, output that is serializable but simply too large to be handled by LLMs efficiently (e.g. large HTML documents) are a highly impractical side-effect of dynamic tool-generation.

In our framework, we tackle this issue by separating the Tool Agent from the immediate consumption of the tool's output. After a tool is executed, we check whether the output is serializable and can be directly used by the agent. If that is not the case, we provide the agent with a handle to the artifact in which the original output is stored. It is pickled, stored to file, and can be reused as input to subsequent tool calls. Crucially, we employ a small LLM to generate a concise description of the output. This description is based on the tool itself, its output and the purpose of the call as provided by the Tool Agent at the time of execution, allowing the agent to understand and reason about the result without needing to process the full raw data. While this process can be cumbersome, it is a first step towards enabling agents to handle incompatible datatypes, an important requirement for explorative situations in which unexpected situations will occur.

Security of Arbitrary Code Execution. One of the more intuitive concerns is the issue of arbitrary code execution. While predefining tools restricts agents capabilities, it also provides a level of security and predictability. With dynamic tool generation, however, agents can, in theory, generate arbitrary code that is potentially harmful, whether done intentionally or by accident. In our implementation, we solely validate whether the tools are executable and scan for unsafe imports based on a minimal heuristic. However, once tool generation is unconstrained, agents could quickly produce code that is unpredictable, introduces subtle errors, or has unintended side effects. Managing these risks is not just a technical problem - it also requires a shift in perspective on autonomy and reliability. In many ways, it's similar to trusting human astronauts: rigorous training and screening reduce risk, but there is never absolute certainty that they will act perfectly. For autonomous agents, pretraining, safety checks, and validation can help, but some uncertainty will always remain. Oversight mechanisms could be added to monitor or constrain agent behavior, yet doing so limits the very flexibility and creativity that make dynamic tool generation valuable. And if the oversight itself becomes AI-driven, the same question remains: how can we ensure that the system enforcing trustworthiness follows its instructions? This layered dilemma underscores that enabling true autonomy in deep-space missions is as much an ethical challenge as it is a technical one.

Fidelity of Generated Tools and Physics Models In the motivating example, the generated tools utilized a linear approximation for trajectory propagation to demonstrate the framework's code generation capabilities. This approximation is valid only for extremely short horizons (seconds), where gravitational curvature is negligible, For operational collision avoidance, higher-fidelity models such as the Clohessy-Wiltshire-Hill (CWH) equations for close-proximity relative motion are required. This highlights a critical dependency: the frameworks output quality is bounded by the agents knowledge of such domain-specific physics. Future iterations will require to integrate domain-informed constraints or domain-specific libraries into the agents context to ensure the generated tools adhere to orbital mechanics standards suitable for flight.

5.3 Implications for Space Missions

Artificial Intelligence (AI) is already being adopted into the space domain. NASA uses AI to enable robots like the *Perseverance* [5] rover to make real-time navigation decisions without Earth's interventions. *CIMON* [6], an interactive assistant for astronauts, was deployed in the Columbus module of the ISS during the Horizons mission [3] in 2018.

NASA explicitly states that autonomous operations are a key capability for deep-space missions, where live communication to and interventions from Earth is impossible [2, 4]. As a result, future space exploration will need to do more than simply follow predefined procedures - they must be able to reason, adapt, and respond to unexpected situations. Current AI systems onboard rovers and robotic assistants, while capable of sophisticated navigation and decision-making, still rely on fixed tools and pre-programmed responses. In contrast, a system based on large language models that can dynamically generate tools opens the door to true autonomous discovery: it could analyze sensor data, formulate hypotheses, and create new computational or operational tools on the fly, without human intervention. For example, when encountering unexpected sensor readings, the system could develop tools for data-filtering or clustering, aiming to identify anomalies in the system readings. This kind of adaptability is especially critical for deep-space missions to outer planets and beyond, where severe communication delays make real-time guidance from Earth impossible. By enabling onboard reasoning, flexible problem-solving, and autonomous tool creation, LLM-based systems could dramatically extend the scientific and operational reach of future missions, turning spacecrafts into explorers capable of handling challenges that humans on Earth cannot anticipate.

We envision the Multi-Agent System operating as an autonomy layer on top of the vehicle's standard systems. The Reasoning Agent does not directly access hardware. Instead, it accesses a shared backboard where subsystems (e.g. power, thermal, navigation, etc.) publish telemetry, In an operational scenario the agent acts proactively: it continuously monitors specific telemetry artifacts for threshold violations and upon detection, it triggers planning workflows.

6 CONCLUSION

We presented a framework enabling autonomous multi-agent systems to dynamically generate and execute Python tools at runtime, addressing deep space communication constraints. Our approach connects high-level reasoning with environment interaction by creating task-specific tools on demand. Central to our approach is separating reasoning from computation. Agents delegate numerical operations to tools, ensuring precise trajectory estimation while the LLM focuses on orchestration. In our space debris example, agents autonomously generated tools for trajectory analysis and closest-approach estimation without predefined instructions. Future work will address challenges in tool output handling, security and integration with spacecraft autonomy concepts. As missions

move further from Earth, dynamic tool generation offers a path to handle unfamiliar situations without predefined functionality.

REFERENCES

[1] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[2] Molly Anderson and Julia Badger. 2024. Gateway Autonomy for Enabling Deep Space Exploration. In *2024 IEEE Aerospace Conference*. 1–7. https://doi.org/10.1109/AERO58975.2024.10521067

[3] Marius Bach and Dieter Sabath. 2018. Horizons Mission-Challenges and Highlights. (2018).

[4] Abigail Bowman. 2025. *Intelligent and Adaptive Systems.* https://www.nasa.gov/ames/core-area-of-expertise-intelligent-and-adaptive-systems/?utm_source=chatgpt.com

[5] Abigail Bowman. 2026. *Artificial Intelligence - Safely using advanced AI tools to support missions and research projects across the agency.* https://www.nasa.gov/artificial-intelligence/

[6] German Aerospace Center (DLR). 2018. *CIMON - the intelligent astronaut assistant.* https://www.dlr.de/en/latest/news/2018/1/20180302_cimon-the-intelligent-astronaut-assistant_26307

[7] Mohd Ariful Haque, Justin Williams, Sunzida Siddique, Md. Hujaifa Islam, Hasmot Ali, Kishor Datta Gupta, and Roy George. 2025. Advanced Tool Learning and Selection System (ATLASS): A Closed-Loop Framework Using LLM. In *2025 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 64–73. https://doi.org/10.1109/SOSE67019.2025.00012

[8] Xu Huang, Junwu Chen, Yuxing Fei, Zhuohan Li, Philippe Schwaller, and Gerbrand Ceder. 2025. CASCADE: Cumulative Agentic Skill Creation through Autonomous Development and Evolution. arXiv:2512.23880 [cs.AI] https://arxiv.org/abs/2512.23880

[9] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. 2024. Qwen2. 5-Coder Technical Report. *arXiv preprint arXiv:2409.12186* (2024).

[10] Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, and Zhicheng Dou. [n.d.]. DeepAgent: A General Reasoning Agent with Scalable Toolsets. https://doi.org/10.48550/arXiv.2510.21618

[11] Yuan Li, Yixuan Zhang, and Lichao Sun. [n.d.]. MetaAgents: Simulating Interactions of Human Behaviors for LLM-based Task-oriented Coordination via Collaborative Generative Agents. http://arxiv.org/pdf/2310.06500v1

[12] ESA Space Debris Office. 2025. *ESA's Annual Space Environment Report.* Technical Report. European Space Agency (ESA).

[13] Gemma Team. 2025. Gemma 3. (2025). https://goo.gle/Gemma3Report

[14] Qwen Team. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[15] Qwen Team. 2025. QwQ-32B: Embracing the Power of Reinforcement Learning. https://qwenlm.github.io/blog/qwq-32b/

[16] Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. 2025. Recursively summarizing enables long-term dialogue memory in large language models. *Neurocomputing* 639 (2025), 130193. https://doi.org/10.1016/j.neucom.2025.130193

[17] Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. 2025. LLM Agents Making Agent Tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, Stroudsburg, PA, USA, 26092–26130. https://doi.org/10.18653/v1/2025.acl-long.1266

[18] Andrea Zollo, Cristina Parigini, Roberto Armellin, Juan Felix San Juan Diaz, Annarita Trombetta, and Ralph Kahle. 2026. A polynomial-based Monte Carlo approach for estimating long-term collision probabilities. *Acta Astronautica* 242 (2026), 178–192.

A MOTIVATING EXAMPLE WALKTHROUGH

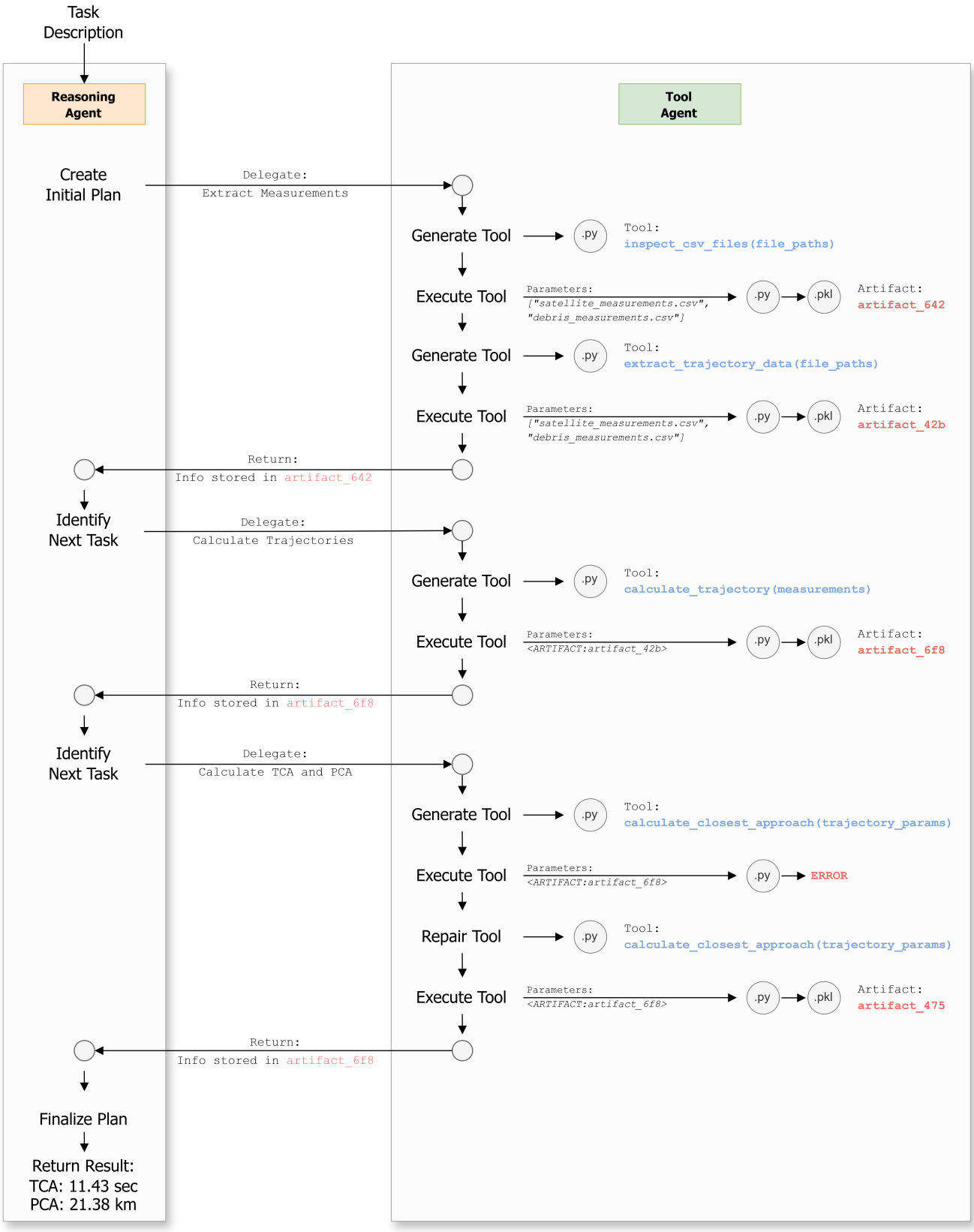


Figure 4: Walkthrough of Collision Assessment

Toward Automated Operational Assist in Satellite Fleet Management via Organizational MARL

Julien Soulé
SEDAN Research Group – SnT – University of Luxembourg
Luxembourg, Luxembourg
julien.soule@hotmail.fr

Abstract

Earth-observation satellite fleets are increasingly difficult to operate with limited mission-control staff in a highly evolving spatial environment while facing mounting challenges such as inherent uncertainty, time constraints, communication disruptions, cyber threats, resource limits, and collision risks. We propose to address this global challenge through an automated operational-assist approach with two complementary contributions: (i) a fully tunable environment that abstracts the main features of ground operations, enabling controlled, reproducible fleet-level experimentation across diverse scenarios; (ii) an explicit decision architecture combining organizational reinforcement learning to improve coordination while preserving safety, controllability, and explainability through trajectory-level analysis, yielding actionable recommendations and partial automation to support operators. Compared with handcrafted, planning-style, and unconstrained learning baselines, our approach improves long-horizon operational performance and safety compliance while providing useful diagnostics for operator audit. These results support a progressive path from simulation benchmarks to supervised operational-assist deployment.

CCS Concepts

• **Computing methodologies** → **Machine learning**.

Keywords

Satellite Fleet Management, Organizational Multi-Agent Reinforcement Learning, Multi-Agent Systems, Explainability

ACM Reference Format:
Julien Soulé. 2026. Toward Automated Operational Assist in Satellite Fleet Management via Organizational MARL. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 9 pages.

1 Introduction

Earth-observation satellite fleets are increasingly difficult to operate with limited mission-control staff in a rapidly evolving space environment [12, 24, 26]. Operational teams must coordinate under uncertainty and time pressure while handling communication disruptions, cyber threats, resource limits, and collision risks. This creates a strong need for automated operational-assist systems that support decision-making without removing human oversight.

Throughout this paper, we use several key concepts. *Operational-assist* refers to decision-support systems that augment human operators without removing their authority. *Organizational constraints* are explicit roles and missions that guide agent behavior beyond reward optimization. *Role specialization* captures differentiation of agent behaviors to improve coordination efficiency. *Mission-phase discipline* describes structured progression between operational phases (acquire, deliver, stabilize) rather than greedy behavior.

Operationally, the target behavior is to maximize useful data delivered to the ground while preserving fleet health. A central tension is that serving observation tasks and delivering value are decoupled phases: agents can be locally productive yet globally ineffective if buffered data is not relayed within time constraints. This acquire–deliver logic motivates explicit coordination.

From a multi-agent perspective, each satellite can be modeled as an autonomous agent interacting with teammates in a partially observable setting. This naturally motivates Dec-POMDP formulations and Multi-Agent Reinforcement Learning (MARL) methods [20, 21, 23, 31]. However, raw MARL performance is insufficient for deployment, since operational practice also requires explicit controllability, traceable safety compliance for operator audit [1, 3, 11].

To address this global challenge, we propose two complementary contributions. First, we introduce *Orbital Resilient Benchmark for Interactive Task-aware Autonomous Learning* (ORBITAL), a tunable environment for controlled, reproducible fleet-level experimentation. Second, we propose a decision-making architecture combining decentralized autonomy, organizational constraints, and human oversight. The core control layer uses organizational reinforcement learning grounded in $\mathcal{M}OISE^{+}$ and MOISE+MARL [15, 16, 28] to improve coordination while preserving safety, controllability, and explainability through trajectory-level analysis.

Compared with handcrafted and unconstrained learning baselines, results indicate that our approach improves long-horizon operational performance and safety compliance, while providing actionable diagnostics for operator audit [28]. These findings support a progressive deployment path from simulation benchmarks to supervised operational-assist deployment.

Section 2 reviews related work; Section 3 presents technical background; Section 4 introduces ORBITAL and the decision-making architecture; Section 5 details the experimental protocol; Section 6 reports results; and Section 7 concludes with a deployment path.

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

2 Related work

Satellite Operations and Decision Support Satellite planning and scheduling have been extensively studied in operations research, including Earth-observation mission planning, agility constraints, and integrated allocation/scheduling formulations [9, 12, 24]. These works provide strong optimization baselines and realistic constraint modeling. However, they usually focus on centrally optimized plans and are less oriented toward adaptive, decentralized, and continuously learning operational-assist loops.

In parallel, agent-based autonomy has long been investigated for spacecraft constellations [26]. This line of work supports distributed decision-making, but often without a unified benchmark that jointly emphasizes modern MARL evaluation, explicit organizational control, and operator-oriented explainability.

Digital Twin and Simulation for Space Systems Recent work on digital twins for space systems highlights their potential for system validation, software testing, and predictive decision support [6, 19]. Nevertheless, many current approaches target specific subsystems or engineering workflows. For fleet-level autonomy research, there remains a need for reproducible environments that are simple enough for controlled MARL experimentation while progressively extensible toward higher realism.

MARL, Safety, and Explainability Cooperative MARL methods have significantly progressed, from actor-critic and value factorization methods to practical training recipes [10, 20, 23, 31]. Benchmarks and software ecosystems such as SMAC, PettingZoo, Gymnasium, and MARLlib improved reproducibility and comparison [14, 25, 29, 30]. However, these environments do not directly represent satellite operational constraints and domain-specific safety priorities.

Safety-aware reinforcement learning (RL) has introduced constrained optimization and shielding methods [1, 3, 11], while explainable RL research has proposed post-hoc and introspective analysis tools [22, 27]. Yet, these strands are often developed separately from organizational modeling and from domain-specific operational-assist requirements.

Organizational Modeling and Organizational MARL Organizational MAS modeling (roles, groups, missions, deontic constraints) is well established through Agent-Group-Role (AGR) and $\mathcal{M}OISE^+$ [8, 15, 16]. Building on this foundation, MOISE+MARL integrates organizational constraints into MARL and uses trajectory-based post-analysis to assess organizational alignment [28]. This direction is promising for controllability and explainability, but it has not yet been fully studied in the context of satellite-fleet operational-assist settings with an explicit benchmarking perspective.

Overall, no single line of work jointly satisfies high operational relevance for satellite fleets, adaptive multi-agent learning, explicit organizational control, and operator-oriented explainability. This motivates our contributions, comprising the operationally grounded benchmark ORBITAL for satellite fleet assistance and the decision-making architecture built on organizationally constrained MARL for safety-aware controllability and trajectory-level analysis for actionable explainability.

3 Background

This section provides the technical background we built on.

3.1 Cooperative Dec-POMDP

We formalize fleet-level decision-making as a Dec-POMDP [4, 21]. This framework is suitable for ORBITAL-like settings where agents act under local observability, decentralized execution, and team-level objectives.

A Dec-POMDP instance is:

$$d = \langle S, \{A_i\}_{i=1}^n, T, R, \{\Omega_i\}_{i=1}^n, O, \gamma \rangle,$$

where S is the latent state space, A_i and Ω_i are local action and observation spaces for agent i, T is the transition kernel, O the observation kernel, R a cooperative reward function, and $\gamma \in [0, 1]$ the discount factor.

Let $\pi_i(a_i \mid \tau_i)$ denote the local policy of agent i over local action-observation history τ_i. The joint policy is $\pi = (\pi_1, \ldots, \pi_n)$ and optimizes expected return:

$$V(\pi) = \mathbb{E}_{\pi,T,O}\left[\sum_{t=0}^{H-1} \gamma^t r_t\right].$$

In ORBITAL, this objective already embeds operational trade-offs (service, energy, communication, resilience), but by itself it does not guarantee role specialization, safety-compliant behavior, or interpretability.

3.2 Organizational modeling with $\mathcal{M}OISE^+$

To encode controllable coordination semantics, we rely on $\mathcal{M}OISE^+$ [15, 16]. Its key contribution is to separate organization into complementary layers:

(1) **Structural specifications**: roles and role relations (specialization or inheritance-like structures).
(2) **Functional specifications**: goals and missions that structure collective progress.
(3) **Deontic specifications**: permissions and obligations linking roles to missions under conditions.

This layered representation is important because it separates *what* should be done (functional), *by whom* (structural), and *under which normative constraints* (deontic), instead of collapsing everything into scalar reward coefficients.

3.3 MOISE+MARL as organizational control layer

MOISE+MARL [28] injects organizational knowledge into MARL while keeping standard MARL backbones usable. The integration relies on three families of guides:

(1) **Role-action guides** (RAG): constrain or prioritize actions based on role and trajectory context.
(2) **Role-reward guides** (RRG): penalize role-inconsistent decisions.
(3) **Goal-reward guides** (GRG): reward mission-consistent progress patterns.

In the following, we use the abbreviations RAG, RRG, and GRG for readability.

For agent i at time t, the effective decision/reward mechanism can be summarized as:

$$a_{i,t} \sim \pi_i(\cdot \mid \tau_{i,t}) \text{ over } \tilde{A}_{i,t} = rag(h_{i,t}, o_{i,t}),$$

$$\tilde{r}_t = r_t + \sum_{m \in \mathcal{M}_{i,t}} grg_m(h_t) + rrg(h_{i,t}, o_{i,t}, a_{i,t}).$$

The result is a hybrid control paradigm in which learning remains data-driven while search is guided by explicit priors. This can reduce unsafe exploration regions and improve policy controllability, particularly in partially observable cooperative settings [5, 28].

3.4 Trajectory-based analysis for explainability

Even if policies are trained with organizational constraints, one still needs post-hoc evidence of the actual learned behavior. The **Trajectory-based Evaluation in MOISE+MARL (TEMM)** method [28] addresses this gap. TEMM operates on multi-episode trajectories and infers implicit organizational regularities. At a high level: i) infer structural regularities (role-like behavior clusters);; ii) infer functional regularities (goal/mission progression patterns);; iii) compare inferred and intended structures to quantify alignment.

This process yields quantitative interpretable artifacts through: $\text{OF} = \alpha \cdot \text{SF} + (1 - \alpha) \cdot \text{FF}$, where structural fit (SF) captures role consistency, functional fit (FF) captures mission/goal consistency, and organizational fit (OF) summarizes both dimensions.

In ORBITAL-like environments, pure return maximization can produce brittle policies that exploit local shortcuts while degrading long-term viability. Typical failure modes include energy collapse, communication fragmentation, or cyber-sensitive behavior such as persisting in observation or relay actions while compromise indicators are high. The combination of Dec-POMDP formalization, organizational constraints, and trajectory-level organizational analysis provides the conceptual foundation needed to evaluate policies not only by performance but also by controllability, safety compliance, and explainability.

4 Method

This section describes our two coupled contributions, namely (i) ORBITAL, an operationally grounded benchmark for satellite fleet operational-assist, and (ii) a decision-making architecture relying on an ORBITAL-oriented MOISE+MARL, which injects organizational control and interpretability into MARL policies.

4.1 The ORBITAL environment

ORBITAL [1] is intentionally designed as a benchmark for *operational-assist* use, not as a full-fidelity orbital propagator. The objective is to keep the environment simple enough for controlled MARL experimentation while preserving the interaction structure that makes fleet management difficult in practice.

ORBITAL offers two geometric modes: a default 2D orbital abstraction with states (θ, r), as shown in Figure 1, and a 3D mode that includes an inclination variable ϕ for enhanced visualization, illustrated in Figure 2. While the 3D mode increases realism, it does not alter the core decision problem of balancing task servicing, delivery, and safety under uncertainty. Thus, the 2D mode is used as the primary view for clearer comparisons across conditions.

Design requirements The benchmark was designed around five requirements derived from our research problem: i) represent **observation-task pressure** (dynamic and priority-sensitive tasks),; ii) represent **resource pressure** (energy-limited long-horizon decisions),; iii) represent **communication uncertainty** (time-varying connectivity),; iv) represent **cyber uncertainty** (degraded sensing/acting/relaying),; v) represent **conjunction-risk pressure** from orbital debris fields. This is why ORBITAL combines non-stationary observation tasks, finite energy with heterogeneous action costs, stochastic communication degradation, stochastic compromise events, and drifting debris clouds that induce local conjunction-risk signals.

4.2 Reference Operational Scenario

We evaluate this architecture on a reference Earth-observation scenario inspired by agile LEO constellation operations. The setup considers eight satellites distributed over three orbital shells, dynamic observation demands with priorities, intermittent downlink opportunities, and coupled safety pressures (energy depletion, communication isolation, cyber degradation, and conjunction-risk proxy from debris density).

The intent is not to replicate a specific industrial mission one-to-one, but to preserve the operational doctrine that matters for assistive autonomy: *acquire–deliver–stabilize*. In this doctrine, observation throughput alone is insufficient if delivery windows are missed or if safety margins are consumed too aggressively.

Table 1: Reference operational scenario (used for all main comparisons).

Aspect	Scenario choice
Constellation geometry	8 cooperative satellites over 3 LEO shells
Mission demand	Non-stationary observation tasks with dynamic priority
Delivery model	Inter-satellite relay and intermittent ground-station windows
Safety pressures	Energy limits, link degradation, compromise events, debris-risk proxy
Success criterion	High delivered value with bounded risk and no mission collapse

Operational state and coupling At time t, the latent state includes satellite positions, energy levels, buffered data, compromise timers, active observation-task set, and communication adjacency matrix. Satellites are coupled through shared observation tasks, shared communication paths to ground, and shared team-level reward. This coupling produces the coordination tension we need to study because individual actions affect both local and fleet-level viability.

Observation and action modeling ORBITAL uses fixed-size local observations (16-dimensional vectors) and a compact discrete action space of size 7 (`Observe`, `Relay`, `OrbitDown`, `OrbitUp`, `LowPower`, `CyberScan`, `Idle`). The fixed vector format supports reproducible MARL pipelines and avoids benchmark bias toward a specific architecture. The action set was chosen to reflect the minimum operational primitives needed for operational-assist settings, including task servicing, data return, mobility, energy management, security response, and fallback behavior. These interface choices are

[1] An implementation of ORBITAL and the conducted experiments, including all details (organizational specifications, hyperparameters, and architectures), are available at https://github.com/julien6/ORBITAL.git.

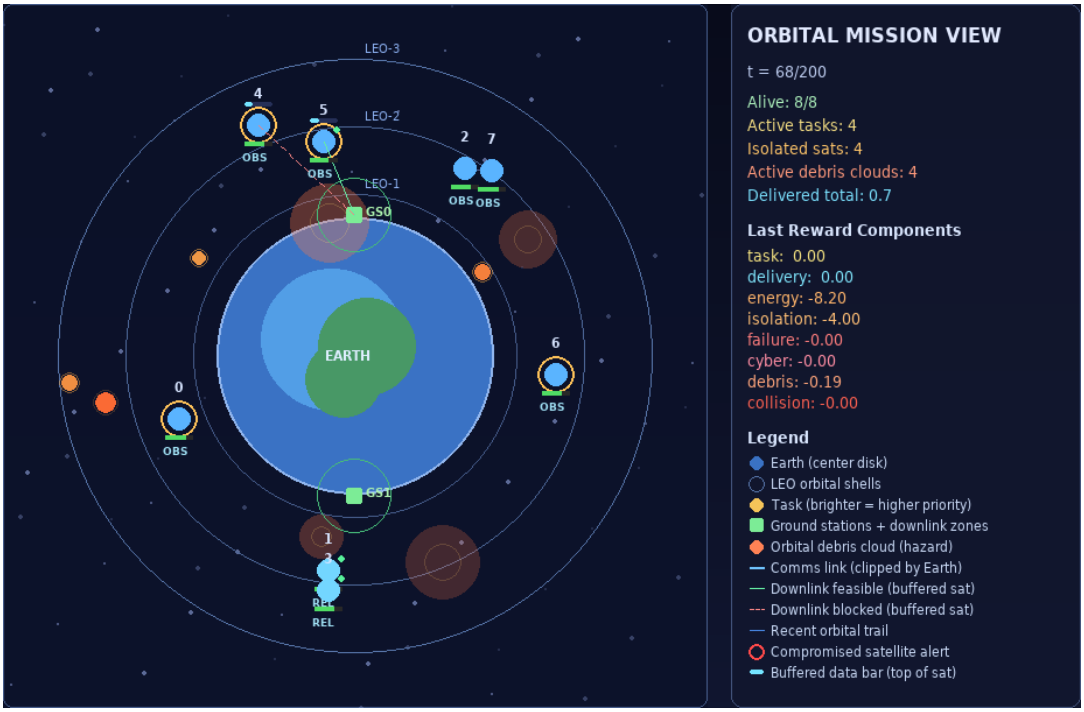


Figure 1: Screenshot of the ORBITAL environment (t = 68/200) in its orbital representation. Earth is shown at the center, surrounded by three low-Earth-orbit (LEO) shells (LEO-1 to LEO-3). Eight cooperative satellites (indexed markers) evolve on orbital states (θ, r), with role tags (`OBS`/`REL`) and per-satellite buffered-data bars. Orange markers denote active observation tasks (brightness proportional to priority). Green squares indicate ground stations (GS0, GS1) with downlink windows. Orange translucent halos denote orbital debris clouds and conjunction-risk zones. Inter-satellite communication links and downlink rays are line-of-sight constrained and clipped by Earth (no signal through Earth); feasible buffered downlinks are shown in green, blocked ones in dashed red. The right panel (*ORBITAL MISSION VIEW*) reports mission/safety indicators (alive satellites, active tasks, isolated satellites, active debris clouds, delivered total) and the per-step reward decomposition (`task`, `delivery`, `energy`, `isolation`, `failure`, `cyber`, `debris`, `collision`), highlighting the trade-off between mission productivity, connectivity, and conjunction-safe resilience.

intentionally minimal: fixed-size vectors improve reproducibility across MARL families, the 7-action set captures core operational primitives, and both Agent Environment Cycle (AEC) and Parallel application programming interfaces (APIs) are available with consistent semantics.

Task-delivery decoupling A critical modeling decision is to separate *observation-task servicing* from *value delivery*. `Observe` converts local task opportunities into buffered data, but mission value is maximized only when data is later relayed through available communication opportunities toward ground. For example, a satellite may continue observing tasks while its buffer is full and no relay opportunity is available. While this maximizes local productivity, it degrades global mission performance since collected data cannot be delivered. In parallel, a local debris-density signal yields a simplified conjunction-risk proxy (*Pc*-like indicator) that can be reduced through orbital maneuver actions (`OrbitDown`/`OrbitUp`). We use this proxy as an operational trigger variable: if risk remains high across successive steps, the architecture should shift from productivity-focused actions toward safety-preserving actions. This separation forces policies to balance sensing throughput, topology management, delivery timing, and collision-risk mitigation rather than greedily optimizing local sensing only.

Reward design for operational trade-offs Default reward mode is shared team reward with positive terms for mission productivity and penalties for resilience degradation: $r_t = w_{\text{task}}\, c_{\text{task}} + w_{\text{delivery}}\, c_{\text{delivery}} - w_{\text{energy}}\, c_{\text{energy}} - w_{\text{isolation}}\, c_{\text{isolation}} - w_{\text{failure}}\, c_{\text{failure}} - w_{\text{cyber}}\, c_{\text{cyber}} - w_{\text{debris}}\, c_{\text{debris\_risk}} - w_{\text{collision}}\, c_{\text{collision}}$.

This reward structure is intentionally non-myopic, since maximizing return requires balancing service, survivability, connectivity, cyber resilience, and conjunction-risk control over the whole horizon. ORBITAL also provides a local reward mode for controlled comparisons.

Episode termination and realism scope Episodes stop at horizon or mission collapse (no alive satellites or critically low survivability). This choice makes unsafe policies self-limiting in long runs. ORBITAL remains a simplified environment, but it preserves the operational couplings required to evaluate coordination doctrine, safety control, and human-supervised assist behavior.

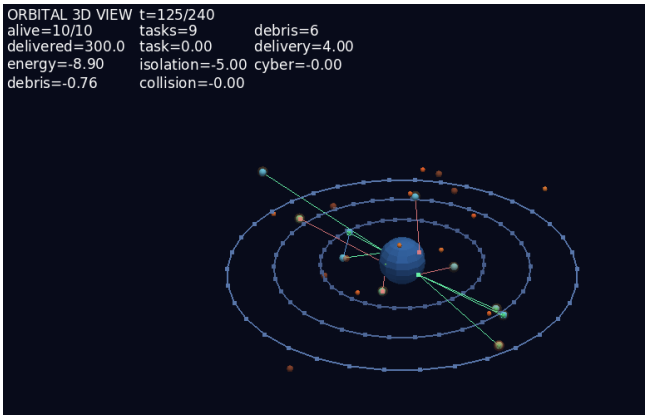


Figure 2: ORBITAL 3D mode screenshot. The same operational entities are represented in 3D (Earth, LEO shells, satellites, ground stations, debris, and communication/downlink links), with the same mission semantics as in the 2D mode.

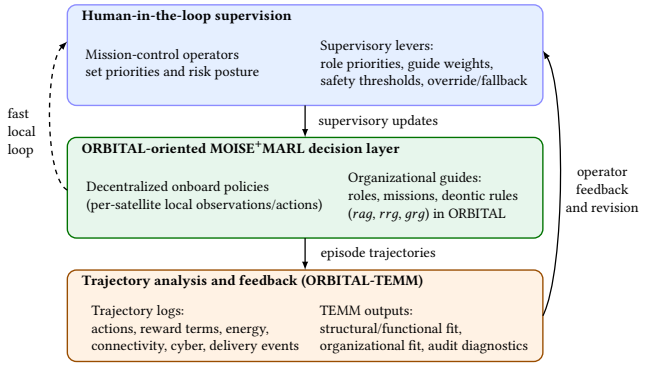


Figure 3: Operational-assist architecture based on an ORBITAL-oriented MOISE$^+$MARL. Human supervision sets priorities and safety posture; decentralized policies are constrained by organizational guides; ORBITAL-TEMM analyzes trajectories and returns audit-oriented feedback for supervisory updates and corrective override.

4.3 ORBITAL-oriented decision-making architecture

In the ORBITAL setting, static rule-based scheduling systems remain easy to inspect and certify. However, they tend to be brittle when facing evolving observation demand, fluctuating inter-satellite connectivity, and stochastic cyber or operational disturbances, as commonly observed in large-scale Earth observation constellations [32]. Distributed Constraint Optimization Problem (DCOP) formulations provide an explicit framework for decentralized coordination and property-preserving optimization in such scenarios [18], enabling safety guarantees.

However, extending DCOP-based approaches to support long-horizon adaptation under partial observability and non-stationary uncertainties typically requires non-trivial model extensions and proactive mechanisms. Adaptation is therefore less direct than in learning-based control [13]. Vanilla MARL is highly adaptive to dynamic environments because it learns directly from interaction data. However, it lacks explicit mechanisms to enforce structured behaviors such as: (i) role specialization (e.g., some satellites focusing on observation while others relay data), and (ii) mission-phase discipline (e.g., prioritizing delivery when buffers are saturated). In such cases, reward shaping alone is often insufficient to ensure consistent and safe behavior [7]. Organizational MARL is therefore adopted here as a middle ground. It enables preserving data-driven adaptation while injecting declarative role and mission constraints that improve system-level controllability, interpretability, and post-hoc auditability of agent behaviors [28].

Therefore, building on the MOISE+MARL [28] framework, we propose an ORBITAL-oriented organizational architecture [1] aligned with mission-control practice (Figure 3).

Architecture overview The architecture is organized into three coupled layers with different timescales. The top layer is human-in-the-loop supervision, where operators set mission priorities and safety posture through high-level levers (role priorities, guide weights, safety thresholds, and fallback policy). The middle layer is decentralized onboard decision-making under organizational constraints (roles, missions, and deontic rules). The bottom layer is trajectory analysis (ORBITAL-TEMM), which produces interpretable diagnostics for audit and corrective feedback.

Two control loops The design features a fast local loop and a slower supervisory loop. The fast loop allows each satellite to act based on local observations and organizational guides, while the supervisory loop updates high-level guidance without direct teleoperation. This structure ensures reactivity and maintains operator authority. For example, if communication quality declines and buffered data increases, supervisors can enhance delivery guidance (prioritizing relay roles), prompting a shift towards relay actions. Conversely, if compromise and isolation indicators rise, supervisors can tighten safety thresholds, encouraging safety-guard behavior and minimizing risky actions. In both scenarios, intervention is high-level, while action selection remains decentralized.

Feedback and auditability Trajectory logs (actions, reward components, energy, connectivity, cyber events, delivery events) are analyzed by ORBITAL-TEMM to estimate structural/functional alignment and organizational fit, then returned to supervision for the next revision cycle.

Organizational mapping We map ORBITAL operational intents to $\mathcal{MOISE}^+$-style specifications:

(1) **roles**: behavioral specialization templates,
(2) **goals**: trajectory-level objectives with measurable progress,
(3) **missions**: coherent groups of goals in operational phases,
(4) **deontic rules**: permissions/obligations linking roles to missions.

This mapping gives a declarative control layer that can be inspected and revised independently from the MARL backbone. Table 2 summarizes the role profile used in this paper.

Constraint guides adapted to ORBITAL The ORBITAL adaptation ties guide logic to operational indicators available in trajectories and observations, including local task opportunity, buffered data

Table 2: ORBITAL-oriented organizational specifications (high-level view).

Role	Main mission focus	Typical obligations/permissions
Observer	Prioritized acquisition	Obliged to observe feasible high-priority tasks; permitted to defer low-priority opportunities under strong resource pressure
Relay	Delivery continuity	Obliged to relay when buffered data and communication opportunities are favorable; permitted to adapt relay cadence under congestion
Safety guard	Fleet survivability	Obliged to prioritize safe actions under elevated risk (energy depletion, isolation, compromise); permitted to override non-critical productivity actions

pressure, communication degree, energy margin, compromise status, and recent delivery events.

Mission semantics We structure behavior around three mission families: i) **Acquisition mission**: increase task servicing under feasibility constraints.; ii) **Delivery mission**: convert buffered data into delivered value reliably.; iii) **Resilience mission**: preserve fleet viability under resource/cyber stress. Role-specific permissions and obligations switch emphasis between these missions according to context. This produces a controllable trade-off between productivity and safety rather than relying on implicit reward.

Hard and soft control regimes The adaptation supports two control regimes. In hard regimes, RAG can enforce action masking so that non-authorized actions are unavailable. In soft regimes, actions remain selectable but violations are discouraged through RRG. This hardness axis is useful to tune exploration freedom versus organizational compliance.

ORBITAL-adapted TEMM We integrate an ORBITAL-specific TEMM process [28] as post-training validation and explainability. Trajectories include action traces, reward components, energy evolution, communication context, task-service and delivery events, and cyber-state transitions. TEMM is applied in three stages: i) infer structural regularities corresponding to implicit role behavior;; ii) infer functional progression patterns corresponding to missions/-goals;; iii) compare inferred and intended specifications to quantify alignment. We report organizational fit as $\mathrm{OF} = \frac{1}{2} \cdot (\mathrm{SF} + \mathrm{FF})$.

5 Experimental setup

This section describes the experimental protocol.

5.1 Experimental goals

We target four experimental questions: i) **Q1 (Performance):** Does ORBITAL-oriented MOISE+MARL improve long-horizon mission execution compared with unconstrained learning and handcrafted coordination?; ii) **Q2 (Control/Safety):** Does organizational guidance reduce unsafe or mission-degrading behavior under resource, communication, and cyber stress?; iii) **Q3 (Robustness):** Does the approach remain effective under intensified non-stationarity (task dynamics, link drops, compromise rate)?; iv) **Q4 (Explainability):** Do TEMM-based organizational indicators provide coherent and actionable post-hoc analysis of learned trajectories?

To answer these questions, each experiment combines: i) one environment configuration (nominal or stressed),; ii) one learning/control condition (baseline or proposed method),; iii) one MARL backbone,; iv) multiple independent random seeds. All protocol parameters and seeds are fixed before result aggregation.

5.2 Hardware and Software Configuration

Hardware profile All experiments were conducted on an academic high-performance computing (HPC) cluster using heterogeneous GPU nodes, including NVIDIA A100, NVIDIA V100, and AMD MI210 devices in a Linux-based scheduling environment. We executed 5 parallel instances per algorithm-environment combination to efficiently explore the baseline and proposed-method parameter spaces while preserving reproducibility through fixed random seeds and deterministic hyperparameter selection on validation data before final evaluation runs.

Software stack The environment is implemented through PettingZoo-compatible APIs (see Table 3). The ORBITAL-oriented decision-making architecture reuses the implementation of MOISE+MARL [2] [28] (organizational specification layer and trajectory analysis), while specializing in role/mission logic for ORBITAL.

Table 3: Software configuration and role in the pipeline.

Component	Role in experiments
Python 3.10	Training and evaluation runtime
Pygame/PyVista	2D and 3D rendering
PettingZoo 1.25.0 [29]	Multi-agent APIs (AEC and Parallel interfaces)
Gymnasium 1.2.3 [30]	Standard RL spaces/wrapping compatibility
PyTorch [17]	Neural policy/value modeling and gradient-based optimization
Optuna [2]	Hyperparameter search and trial-based configuration selection
NumPy 2.2.6	Numerical operations and logging
ORBITAL codebase	Environment dynamics, reward components, rendering/debugging
MOISE+MARL [28]	Organizational guides, role/mission constraints, TEMM pipeline

5.3 Baselines and comparison conditions

Handcrafted baselines To represent operational heuristics, we include: i) **RB-Rule:** rule-based action selection prioritizing local high-priority tasks, then relay, with simple low-energy fallback.; ii) **RB-Relay-heavy:** heuristic emphasizing delivery continuity (relay when possible, observe otherwise), with weak safety adaptation.; iii) **PB-DCOP-lite:** inspired by DCOP decomposition, with periodic assignment of observer/relay intents under communication and energy constraints, then local heuristic execution. These baselines provide interpretable references as operational scripts.

Learning baselines We compare the proposed organizational framework to unconstrained MARL and partially constrained variants: i) **LB-Unconstrained:** same MARL backbone, ORBITAL reward only, no organizational guides.; ii) **LB-RewardOnly:** unconstrained action space, additional reward shaping but no role-action masking.; iii) **LB-ActionOnly:** role-action guidance active, no mission reward guides.; iv) **Proposed (ORBITAL-$\mathcal{M}OISE^+$ MARL):** role-action + role-penalty + mission guides with deontic assignments. This decomposition isolates where gains come from, whether through action control, reward structure, or full organizational coupling.

[2] Accessible at: https://github.com/julien6/MOISE-MARL.

Backbone algorithms To reduce algorithm-specific bias, each condition is evaluated with representative cooperative MARL families: i) actor-critic / policy-gradient style methods (MAPPO [31]),; ii) value-factorization methods (QMIX [23]),; iii) multi-agent actor-critic references (MADDPG/COMA style [10, 20]).

5.4 Ablation plan

To validate causal contributions of the proposed framework, we define the ablations in Table 4. We also stress-test each setting along controlled perturbation axes: i) increased task non-stationarity (spawn/priority volatility),; ii) increased communication degradation ($p_{\text{link\_drop}}$),; iii) increased compromise intensity (adversarial rate and duration),; iv) reduced energy budgets.

Table 4: Ablation settings for ORBITAL-oriented MOISE+MARL.

ID	Ablation description
A0	Full framework (role-action + role-penalty + mission guides + TEMM analysis)
A1	Remove role-action guide (*rag* off): no action-space organizational control
A2	Remove role-penalty guide (*rrg* off): no explicit role-violation penalty
A3	Remove mission guides (*grg* off): no mission-level shaping
A4	Soft-only control: no action masking, penalties/rewards only
A5	Hard-only control: action masking active, no additional role penalty
A6	TEMM feature reduction: remove cyber/context features in trajectory analysis

5.5 Evaluation metrics

We report control, and organizational interpretability metrics.

Mission performance metrics i) **Cumulative return**: episode return.; ii) **Task service volume**: serviced task-priority mass.; iii) **Delivery volume**: delivered data.; iv) **Mission completion rate**: episodes without mission collapse.

Safety and resilience metrics i) **Energy stress index**: low-energy occupancy over agents and time.; ii) **Isolation ratio**: alive agents with zero communication degree.; iii) **Failure count**: depleted satellites per episode.; iv) **Cyber impact score**: aggregate compromise-related penalties/events.

Control and explainability metrics i) **Constraint violation rate**: role-inconsistent action frequency.; ii) **Structural fit** and **functional fit** from ORBITAL-TEMM.; iii) **Organizational fit** using structural and functional organizational fits.; iv) **Consistency score**: intended vs inferred role/mission agreement.

Subjective operator-audit indicators Operator-facing indicators use an ordinal scale (`very low`, `low`, `high`, `very high`): i) **Mission alignment rating**: alignment between inferred motifs and expected mission logic.; ii) **Human-audit agreement rating**: agreement between TEMM diagnostics and human audit.

5.6 Training and evaluation protocol

We use disjoint seeds for training/selection/final evaluation and report aggregate metrics over at least 10 seeds per condition. Learning conditions share identical episode horizons and optimization budgets; hyperparameters are tuned on validation seeds only, then frozen for testing. We report mean, standard deviation, and results uncertainty intervals, with two-sided significance testing and effect sizes for pairwise comparisons. All runs log per-step reward components and mission/safety/organizational events, enabling targeted failure-mode analysis and reproducible reruns.

6 Results and discussion

We first compare strong, medium, and weak organizational-control regimes (Table 5). Strong constraints deliver the fastest early convergence, weak constraints preserve flexibility but slow learning and increase violations, and medium constraints provide the best global compromise. Medium control reaches the highest final return (379.8 ± 16.1), mission success (0.91 ± 0.03), and OF score (0.92 ± 0.02), while keeping convergence substantially better than weak control.

Relating these results to the gaps identified in Section 2, our contributions strongly cover adaptive multi-agent learning, explicit organizational control, and operator-oriented explainability at benchmark level: compared with handcrafted and unconstrained baselines, the proposed framework improves mission value and success, reduces role-inconsistent behavior, and increases trajectory-level interpretability/audit agreement. Coverage of the operational-realism gap is partial by design: ORBITAL captures the key coupled pressures required for operational-assist evaluation, but does not yet represent full flight-grade orbital and communication realism.

6.1 Baselines against handcrafted and learning conditions

Table 6 compares our method against handcrafted and learning baselines. As expected, handcrafted policies do not rely on organizational constraints and therefore violation metrics are marked `n/a`. Two implications stand out. First, handcrafted and planning-style policies remain interpretable and operationally plausible, but plateau at lower performance and resilience levels than the proposed framework. Second, partial organizational variants improve either control or speed, but the full framework is required to simultaneously optimize mission value, convergence, and robustness.

6.2 Convergence and robustness trade-off

The key behavioral result is a non-monotonic relation between control hardness and end-task quality. Overly hard constraints accelerate convergence but reduce robustness; overly weak constraints preserve robustness but underuse organizational priors. Medium constraints act as the best compromise, improving convergence and compliance without collapsing policy diversity.

6.3 Ablation results

Table 7 shows that each component contributes to at least one critical dimension. Removing RAG strongly reduces return and robustness, and increases violations. Removing GRG yields the largest mission-performance drop (-10.4% return). Removing RRG most strongly harms compliance, with the largest violation increase. Reducing TEMM features barely affects training performance but degrades interpretability quality, consistent with TEMM.

6.4 Explainability and trajectory analysis

Quantitative explainability indicators Table 8 shows that the proposed method achieves higher role-cluster separability (0.67 ± 0.04)

Table 5: Aggregate results by constraint regime (mean ± std over seeds). Higher is better for all metrics except convergence episode, violation rate, and energy stress.

Regime	Final return	Return AUC (early)	Convergence episode ↓	Robustness score	Constraint violation ↓	OF score	Mission success rate	Energy stress ↓
Strong constraints	352.6 ± 18.4	205.3 ± 9.8	118 ± 14	0.71 ± 0.05	1.8% ± 0.9	0.90 ± 0.03	0.86 ± 0.04	0.37 ± 0.06
Medium constraints	**379.8 ± 16.1**	192.7 ± 8.2	146 ± 17	**0.84 ± 0.04**	3.9% ± 1.2	**0.92 ± 0.02**	**0.91 ± 0.03**	**0.31 ± 0.05**
Weak constraints	334.9 ± 21.7	143.5 ± 11.6	213 ± 22	0.87 ± 0.04	8.7% ± 1.8	0.81 ± 0.05	0.83 ± 0.05	0.35 ± 0.07

Table 6: Baseline comparison. Handcrafted baselines report `n/a` for organizational violation metrics.

Condition	Final return	Convergence episode ↓	Robustness score	Violation rate ↓	OF score	Delivery volume	Mission success rate
RB-Rule (priority-first)	241.3 ± 12.9	n/a	0.63 ± 0.06	n/a	0.49 ± 0.07	112.4 ± 9.8	0.64 ± 0.08
RB-Relay-heavy	228.1 ± 14.7	n/a	0.67 ± 0.07	n/a	0.45 ± 0.08	121.7 ± 10.3	0.61 ± 0.07
PB-DCOP-lite	286.2 ± 15.8	n/a	0.73 ± 0.06	n/a	0.61 ± 0.06	138.9 ± 9.5	0.72 ± 0.06
LB-Unconstrained	319.7 ± 20.5	229 ± 24	0.82 ± 0.05	10.4% ± 2.1	0.74 ± 0.05	151.8 ± 11.7	0.79 ± 0.05
LB-RewardOnly	341.5 ± 19.2	188 ± 20	0.79 ± 0.05	7.1% ± 1.9	0.82 ± 0.04	162.3 ± 10.8	0.84 ± 0.04
LB-ActionOnly	348.0 ± 17.9	161 ± 18	0.76 ± 0.06	4.6% ± 1.5	0.85 ± 0.04	167.9 ± 10.1	0.86 ± 0.04
Proposed (medium constraints)	**379.8 ± 16.1**	**146 ± 17**	**0.84 ± 0.04**	3.9% ± 1.2	**0.92 ± 0.02**	**182.5 ± 9.4**	**0.91 ± 0.03**

Table 7: Ablation outcomes relative to full method.

Ablation	Δ Return	Δ Robustness	Δ Violation	Δ OF
A1 (no *rag*)	−8.6%	−6.2%	+2.7 pts	−0.07
A2 (no *rrg*)	−3.9%	−4.8%	+3.1 pts	−0.09
A3 (no *grg*)	−10.4%	−2.0%	+0.8 pts	−0.06
A4 (soft-only)	−4.7%	+1.1%	+1.9 pts	−0.05
A5 (hard-only)	−5.3%	−7.4%	−1.4 pts	−0.04
A6 (reduced TEMM features)	−0.2%	−0.1%	+0.0 pts	−0.11

compared to unconstrained learning (0.41 ± 0.06), indicating that organizational constraints enhance behavioral specialization. Audit-agreement ratings improve from `low` to `very high`, reflecting that TEMM patterns align closely with intended specifications.

Table 8: Explainability indicators.

Condition	Role-cluster separability	Mission alignment rating	Human-audit agreement rating
LB-Unconstrained	0.41 ± 0.06	`low`	`low`
LB-RewardOnly	0.49 ± 0.05	`high`	`high`
Proposed (medium)	**0.67 ± 0.04**	`very high`	`very high`

Qualitative and operational relevance Trajectory analysis confirms role/mission motifs in the proposed setting (observer acquisition-to-relay transitions, relay draining patterns, and safety-oriented stress responses), consistent with higher role-cluster separability in Table 8. TEMM outputs provide actionable traceability of role consistency and repeated deontic violations, with mission-alignment and audit-agreement ratings improving from `low` to `very high`.

6.5 Human-in-the-loop audit case

To evaluate auditability, we analyze one representative stress episode where communication degradation and compromise overlap. In the unconstrained condition, two satellites persist in `Observe` despite rising isolation and energy stress, which leads to buffered-data saturation and delayed delivery recovery. TEMM flags this pattern as repeated mission-role mismatch for relay-assigned agents.

Applying a supervised intervention changes the subsequent trajectory: one satellite transitions to relay behavior, one to safety-guard behavior, isolation duration is shortened, and delivery recovery occurs before mission-collapse threshold. Those observations confirmed the intended usage mode of the architecture, where the learning policy remains autonomous between updates, but operators retain structured levers for corrective control.

7 Conclusion

This paper introduced an operational-assist framework for satellite fleet management comprising: i) the ORBITAL environment as an operationally grounded abstraction in a reproducible multi-agent setting under various constraints; ii) and an ORBITAL-oriented decision-making architecture adapted from MOISE+MARL that combines organizational role/mission constraints with trajectory-based analysis to improve controllability and explainability.

The main result is a structured control trade-off: strong constraints accelerate early convergence but may reduce robustness, weak constraints preserve robustness but slow learning, and medium constraints provide the best compromise between performance, convergence speed, and resilience, suggesting that organizational specifications are most useful when guiding safety-critical behavior while preserving policy freedom for emergent cooperation.

However, ORBITAL is a simplified abstraction that lacks full communication, high-fidelity orbital mechanics, and operational complexity. The manually designed organizational specifications may introduce bias and limit scalability. TEMM analysis quality relies on trajectory features and clustering choices, necessitating careful calibration. This simplification may miss second-order effects like communication latency and orbital perturbations. Therefore, we identify three key directions: enhancing environment realism for better simulation of real-world conditions, integrating model-based reinforcement learning and multi-agent world models for improved long-horizon robustness, and reinforcing neuro-symbolic integration to align learned behaviors with safety constraints. Our goal is ultimately developing offline advisory support systems.

Acknowledgments

This DefenceTech project benefits from shared financial support by the Ministry of Foreign and European Affairs, Directorate of Defence, the Ministry of Economy and the Luxembourg National Research Fund (FNR) (DEFENCE24/IS/19272253/ALIAS).

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning*. 22–31.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. arXiv:1907.10902 [cs.LG] https://arxiv.org/abs/1907.10902

[3] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe Reinforcement Learning via Shielding. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). https://doi.org/10.1609/aaai.v32i1.11797

[4] Aurélie Beynier, François Charpillet, Daniel Szer, and Abdel-Illah Mouaddib. 2013. *DEC-MDP/POMDP*. John Wiley & Sons, Ltd, Chapter 9, 277–318. https://doi.org/10.1002/9781118557426.ch9

[5] Jiajun Chai, Zijie Zhao, Yuanheng Zhu, and Dongbin Zhao. 2025. A Survey of Cooperative Multi-Agent Reinforcement Learning for Multi-Task Scenarios. *Artificial Intelligence Science and Engineering* 1, 2 (2025), 98–121. https://doi.org/10.23919/AISE.2025.000008

[6] Andrea Colagrossi, Stefano Silvestrini, Andrea Brandonisio, and Michèle Lavagna. 2026. A Digital Twin Approach for Spacecraft On-Board Software Development and Testing. *Aerospace* 13, 1 (2026). https://doi.org/10.3390/aerospace13010055

[7] Sam Devlin and Daniel Kudenko. 2016. Plan-Based Reward Shaping for Multi-Agent Reinforcement Learning. *The Knowledge Engineering Review* 31, 1 (2016), 44–58. https://doi.org/10.1017/S0269888915000181

[8] Jacques Ferber, Olivier Gutknecht, and Fabien Michel. 2003. Agent/Group/Roles: Simulating with Organizations. In *ABS 2003 - 4th International Workshop on Agent-Based Simulation*, J.P. Muller (Ed.). Montpellier, France. https://hal-lirmm.ccsd.cnrs.fr/lirmm-00269714

[9] Benedetta Ferrari, Jean-François Cordeau, Maxence Delorme, Manuel Iori, and Roberto Orosei. 2025. Satellite Scheduling Problems: A survey of applications in Earth and outer space observation. *Computers & Operations Research* 173 (2025), 106875. https://doi.org/10.1016/j.cor.2024.106875

[10] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (Apr. 2018). https://doi.org/10.1609/aaai.v32i1.11794

[11] Javier Garcia and Fernando Fernandez. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16, 42 (2015), 1437–1480. http://jmlr.org/papers/v16/garcia15a.html

[12] Lorenzo Govoni, Corrado Chiatante, Bjørn Andreas Kristiansen, Tor Arne Johansen, and Andrea Cristofaro. 2026. Optimization-Based Task Allocation for Earth Observation in Multi-Satellite Systems. *Aerospace Science and Technology* 168 (2026), 111058. https://doi.org/10.1016/j.ast.2025.111058

[13] Khoi D. Hoang, Ferdinando Fioretto, Ping Hou, William Yeoh, Makoto Yokoo, and Roie Zivan. 2022. Proactive Dynamic Distributed Constraint Optimization Problems. *Journal of Artificial Intelligence Research* 74 (2022), 179–225.

[14] Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Xiaodan Liang, Zhihui Li, Xiaojun Chang, and Yaodong Yang. 2023. MARLlib: A Scalable and Efficient Multi-agent Reinforcement Learning Library. *Journal of Machine Learning Research* 24, 315 (2023), 1–23. http://jmlr.org/papers/v24/23-0378.html

[15] Jomi Fred Hübner, Jaime Simão Sichman, and Olivier Boissier. 2002. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. In *Advances in Artificial Intelligence*, Guilherme Bittencourt and Geber L. Ramalho (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 118–128.

[16] Hubner, Jomi F et. al. 2007. Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. *Int. Journal of Agent-Oriented Software Engineering* (2007), 370. https://doi.org/10.1504/ijaose.2007.016266

[17] Sagar Imambi, Kolla Bhanu Prakash, and G. R. Kanagachidambaresan. 2021. *PyTorch*. Springer International Publishing, Cham, 87–104. https://doi.org/10.1007/978-3-030-57077-4_10

[18] Shai Krigman, Tal Grinshpoun, and Lihi Dery. 2024. Scheduling of Earth Observing Satellites Using Distributed Constraint Optimization. *Journal of Scheduling* 27, 5 (2024), 507–524. https://doi.org/10.1007/s10951-024-00816-x

[19] Wei Liu, Mengwei Wu, Gang Wan, and Minyi Xu. 2024. Digital Twin of Space Environment: Development, Challenges, Applications, and Future Outlook. *Remote Sensing* 16, 16 (2024). https://doi.org/10.3390/rs16163023

[20] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Advances in Neural Information Processing Systems* 30 (2017).

[21] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer. https://link.springer.com/book/10.1007/978-3-319-28929-8

[22] Erika Puiutta and Eric M. S. P. Veith. 2020. Explainable Reinforcement Learning: A Survey. Springer-Verlag, Berlin, Heidelberg, 77–95. https://doi.org/10.1007/978-3-030-57321-8_5

[23] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 4295–4304. https://proceedings.mlr.press/v80/rashid18a.html

[24] Yure Rocha, Guilherme O. Chagas, Leandro C. Coelho, and Anand Subramanian. 2025. The integrated agile Earth observation satellite scheduling problem. *Computers & Operations Research* 184 (2025), 107212. https://doi.org/10.1016/j.cor.2025.107212

[25] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. *CoRR* abs/1902.04043 (2019).

[26] Thomas Schetter, Mark Campbell, and Derek Surka. 2000. Multiple Agent-Based Autonomy for Satellite Constellations. In *Agent Systems, Mobile Agents, and Applications*, David Kotz and Friedemann Mattern (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 151–165.

[27] Pedro Sequeira and Melinda Gervasio. 2020. Interestingness elements for explainable reinforcement learning: Understanding agents' capabilities and limitations. *Artificial Intelligence* 288 (2020), 103367. https://doi.org/10.1016/j.artint.2020.103367

[28] Julien Soulé, Jean-Paul Jamont, Michel Occello, Louis-Marie Traonouez, and Paul Théron. 2025. An Organizationally-Oriented Approach to Enhancing Explainability and Control in Multi-Agent Reinforcement Learning. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (**AAMAS** 2025)* (Detroit, MI, USA). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1968–1976.

[29] J K Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall Williams, Yashas Lokesh, and Praveen Ravi. 2021. PettingZoo: Gym for Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 15032–15043. https://proceedings.neurips.cc/paper_files/paper/2021/file/803f7c4c3ff61b71be53a0c803bfb57f-Paper.pdf

[30] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. 2025. Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv:2407.17032 [cs.LG] https://arxiv.org/abs/2407.17032

[31] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and YI WU. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 24611–24624. https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf

[32] Itai Zilberstein and Steve Chien. 2026. Large-Scale Continual Scheduling and Execution for Dynamic Distributed Satellite Constellation Observation Allocation. arXiv:2601.06188 [cs.AI] https://arxiv.org/abs/2601.06188

Large-Scale Continual Scheduling and Execution for Dynamic Distributed Satellite Constellation Observation Allocation

Itai Zilberstein
Carnegie Mellon University
Jet Propulsion Laboratory, California Institute of Technology
izilbers@cs.cmu.edu

Steve Chien
Jet Propulsion Laboratory, California Institute of Technology
steve.a.chien@nasa.jpl.gov

ABSTRACT

The size and capabilities of Earth-observing satellite constellations are rapidly increasing. Leveraging distributed onboard control, we can enable novel time-sensitive measurements and responses. However, deploying autonomy to satellites requires efficient computation and communication. This work tackles the challenge of efficiently scheduling observations for hundreds of satellites in a dynamic, large-scale problem with millions of variables. We present the *Dynamic Multi-Satellite Constellation Observation Scheduling Problem* (DCOSP), a new formulation of Dynamic Distributed Constraint Optimization Problems (DDCOP) that models integrated scheduling and execution. DCOSP has a novel optimality condition for which we construct an omniscient offline algorithm for its computation. We also present the *Dynamic Incremental Neighborhood Stochastic Search algorithm* (D-NSS), an incomplete online decomposition-based DDCOP algorithm that repairs and solves sub-problems when problem dynamics occur. We show through simulation that D-NSS converges to near-optimal solutions and outperforms DDCOP baselines in terms of solution quality, computation time, and message volume. As part of the NASA FAME mission, DCOSP and D-NSS will be the foundation of the largest in-space demonstration of distributed multi-agent AI to date.

KEYWORDS

Distributed constraint optimization, scheduling, satellite operations

ACM Reference Format:
Itai Zilberstein and Steve Chien. 2026. Large-Scale Continual Scheduling and Execution for Dynamic Distributed Satellite Constellation Observation Allocation. In *Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., Paphos, Cyprus, May 2026*, IFAAMAS, 10 pages.

1 INTRODUCTION

There has been a proliferation of Earth-observing spacecraft in recent years, including advancements in their capabilities to act as autonomous agents. Reduced launch costs have led to constellations composed of hundreds or thousands of spacecraft that can monitor Earth phenomena [31]. Large observation systems result in shorter revisit times to the target observation locations. Reduced revisit times are crucial for rapid responses to dynamic events such as natural disasters. New spacecraft also possess edge hardware capable of performing more intensive computation onboard, including neural network execution and even planning [7, 8, 10, 40, 51]. The advancement of inter-satellite links (ISL) has enabled persistent communications between spacecraft and stations on the ground.

These capabilities support observation campaigns that require time-sensitive and coordinated measurements. An example is global monitoring of all volcanic activity or flooding events. However, without consistent observation and the ability to react to dynamic events, key measurements of these processes may be missed. For example, wildfire monitoring requires 30-minute updates to be useful to ground responders [23]. Centralized scheduling on the ground suffers from latency and may not be able to meet these timing constraints. Therefore, to successfully tackle these campaigns, we require large-scale observation systems with reduced revisit times and distributed autonomy so that agents can operate online without ground control.

This work focuses on distributed online scheduling that can efficiently coordinate the actions of hundreds or thousands of spacecraft continually as problem dynamics change. Coordinating the actions of a large-scale constellation requires reasoning about agents with varying capabilities, constraints, and visibility of Earth targets while managing limited computational resources. Satellites share CPU and RAM with other critical flight software. Large volumes of communication in space are also unreliable and may even carry a financial cost [25]. These limitations make solving a static problem challenging. However, any operational solution must solve dynamic problems that change over time, which further increases the complexity. In these scenarios, the satellite constellation is fixed, yet the observation requests change. These changing observation requests alter the constraints of the problem. We desire continual scheduling solutions that are lightweight yet can effectively handle these problem dynamics.

Operational satellite observation scheduling uses centralized paradigms [43], and many research efforts have concentrated on centralized solutions [3, 30]. Centralized approaches may be insufficient for dynamic situations that require real-time response due to latencies to the ground. Centralized approaches are also vulnerable to single-point failures that would result in a non-operational constellation.

We model a large-scale satellite constellation as a multi-agent system (MAS) and focus on continual decentralized scheduling that optimizes observation completion during an overlapping scheduling and execution horizon. We present the *Dynamic Multi-Satellite*

Appears at the International Workshop on Autonomous Agents and Multi-Agent Systems for Space Applications (MASSpace-26). Held as part of the Workshops at the 25th International Conference on Autonomous Agents and Multiagent Systems., S. Chien, G. Picard, I. Zilberstein (Chairs), May 2026, Paphos, Cyprus. © 2026 Copyright held by the owner/author(s).

Constellation Observation Scheduling Problem (DCOSP), which is a dynamic distributed constraint optimization problem (DDCOP). DCOSP is a novel application of a DDCOP for dynamic satellite observation scheduling, and extends DDCOPs in multiple ways. The assumptions of DCOSP differ from those of a DDCOP to reflect real-world constraints. We assume agents are aware of the existence of other agents, but do not have access to detailed capabilities or state information. This assumption allows our approach to apply to scenarios with intermittent connectivity, limited bandwidth, or security-mandated communication restrictions. We also formulate a novel optimality condition for a DDCOP that takes into account integrated scheduling and execution. In addition, DDCOP solutions tend to rely heavily on computation and communication, especially when problem dynamics are volatile, which prevents application to DCOSP. These challenges make current DDCOP solvers insufficient for DCOSP.

DCOSP problem instances are much larger than typical DDCOP problems examined in previous literature. DCOSP instances consist of millions of variables that change over time. Agents must react to problem dynamics while expending limited time, memory, and communication. We extend the *Neighborhood Stochastic Search* algorithm (NSS) to a dynamic variation referred to as *Dynamic Incremental Neighborhood Stochastic Search* (D-NSS). NSS decomposes the global problem into smaller sub-problems, producing global solutions more efficiently. D-NSS uses the same foundation and leverages repairing previous solutions to efficiently handle problem dynamics.

The contributions of this work are

(1) a novel formulation of the real-world application of dynamic satellite scheduling as a DDCOP with a unique optimality condition that models task execution,
(2) constructing an omniscient, offline optimal solution to the dynamic satellite scheduling problem, and
(3) presenting the *Dynamic Incremental Neighborhood Stochastic Search algorithm*, a scalable incomplete DDCOP approach.

We evaluate the effectiveness of the approaches on large-scale real-world scenarios as well as analyze the challenges of deploying existing DDCOP solutions to this problem. DCOSP and D-NSS will be leveraged in the largest in-space demonstration of multi-agent AI to date, beginning in 2026. The NASA FAME demonstration involves over 60 participating spacecraft that will dynamically coordinate to observe Earth phenomena [9].

A version of this work appears as an extended abstract in the proceedings of AAMAS 2026 [52].

2 RELATED WORK

2.1 Satellite Observation Scheduling

Satellite observation scheduling is typically framed as an optimization problem that involves geometric reasoning, downlink scheduling, and constraint-based task allocation. The majority of research efforts and operational work have focused on centralized solutions to satellite observation scheduling [1, 3, 6, 11, 17, 18, 30, 43–45, 47]. There is limited work on decentralized scheduling approaches, and these mainly focus on static problems. Examples include auction-based methods [36, 37] and heuristic search-based methods [4, 5, 33, 53]. The work of Zilberstein, Rao, Salis, and Chien proposed the *Multi-Satellite Constellation Observation Scheduling Problem* (COSP), a DCOP formulation of observation allocation [53]. We extend this work by formulating the *Dynamic Multi-Satellite Constellation Observation Scheduling Problem* (DCOSP), which is a novel DDCOP formulation of the problem.

2.2 Dynamic Distributed Constraint Optimization

Distributed constraint optimization problems (DCOP) have modeled applications including mobile sensor teams [34], smart grids [14], and satellite scheduling [53]. Solutions to distributed constraint optimization problems tend to be intensive in computation and communication, making deployment to agents with limited computation challenging. Optimal solutions have exponential complexities [16, 19, 28, 29, 35]. Incomplete DCOP algorithms are more efficient, yet typically rely on agents communicating with all neighboring agents in the constraint graph, resulting in large complexities when constraint graphs are fully connected [26, 32, 46, 49]. The *Neighborhood Stochastic Search* (NSS) algorithm, which iteratively improves sub-problem solutions, has been shown to solve large-scale distributed satellite observation scheduling with limited computation and communication [53]. We extend the NSS algorithm to the *dynamic* DCOP setting (DDCOP) to perform scalable and effective dynamic observation scheduling.

Dynamic distributed constraint optimization problems (DDCOP) [24] extend DCOPs to capture problem changes. A standard DDCOP is composed of a sequence of T static DCOPs where an optimal solution is obtained by solving each of the T DCOPs optimally. DDCOP solutions are inherently online algorithms as a system reacts to changes. Most work has adapted common DCOP algorithms to dynamic variations that inherit computational complexities [2, 12, 22, 27, 41, 48, 54].

When it comes to the application of DDCOPs to satellite operations, there are limitations with the standard definition. When solving static DCOPs, it is possible to assume that solutions are found prior to the execution horizon. However, DDCOPs cannot always make this assumption as dynamics may occur during the execution horizon. This is true for satellite operations; utility is obtained by taking an observation, not scheduling one. Therefore, when the problem changes so that a requested task is removed from the problem, having it scheduled prior to execution may not be optimal. In addition, due to consumptive resources, the starting state of subsequent DCOPs is driven by previous solutions.

3 PROBLEM DEFINITION

3.1 Multi-Satellite Constellation Observation Scheduling

We outline the *Multi-Satellite Constellation Observation Scheduling Problem* (COSP) and discuss the related challenges. COSP is defined by the following sets.

- $H = [h_s, h_e]$: the scheduling horizon.
- A: the set of agents in which each agent is a satellite in the constellation.
- $\mathcal{T}$: the set of point targets on Earth defined by a latitude and longitude.

- R: the set of requests where each request is defined by the target to observe, $\tau \in \mathcal{T}$, and when in the scheduling horizon to observe, $h \subset H$. Note that we index elements of a request r, such as the horizon, with the notation $h(r)$ and use this notation consistently for other variables.
- $S = \bigcup_{a \in A} S_a$: the set of tasks (also referred to as observations) where each S_a corresponds to the tasks of agent a. A task $s \in S_a$ is defined by the request being satisfied, $r \in R$, the interval required to schedule the task, $h \subset h(r)$, and the data volume required to take the observation, $m \in \mathbb{R}^+$.
- $\mathcal{X} = \bigcup_{a \in A} \mathcal{X}_a$: the set of Boolean decision variables where each $\mathcal{X}_a$ corresponds to the variables of agent a. For each $s \in S_a$ we define the Boolean decision variable $x \in \mathcal{X}_a$ where $x = 1$ iff agent a schedules task s.
- $D = \bigcup_{a \in A} D_a$: the set of downlinks where each D_a corresponds to the downlinks of agent a. A downlink is defined by the maximum data volume downlinked, $m \in \mathbb{R}^+$, and the time interval of the downlink, $h \subset H$. We assume that all downlinks are mandatory.
- $C = \bigcup_{a \in A} C_a$: the set of constraints for each agent. Each agent is constrained by processing and data volume. An agent cannot execute two tasks at once and tasks cannot overlap with downlinks. An agent must also never exceed its memory capacity and all observations acquired must be downlinked at the earliest opportunity. Formally,

  $$C_a = C_{D_a} \cup C_{S_a}.$$

  We define

  $$C_{D_a} = \bigcup_{d \in D_a} c_d$$

  where

  $$c_d = \sum_{s \in S_a^d} x(s) \cdot m(s) \leqslant \text{MIN}(m(a), m(d)).$$

  The set S_a^d contains the possible tasks for which the soonest downlink window in the future is d. The value $m(a)$ denotes the memory capacity of agent a. We define

  $$C_{S_a} = \bigcup_{s,s' \in S_a} c_{s,s'}$$

  where

  $$c_{s,s'} = \left[x(s) \cdot x(s') + \mathbb{I}(h(s) \cap h(s') \neq \varnothing) \leqslant 1\right].$$

The objective of COSP is to maximize the number of requests satisfied subject to the constraints. A request is satisfied if a single observation for that request is completed. An optimal assignment of variables $\mathcal{X}^*$ is defined as

$$\mathcal{X}^* = \arg\max_{\mathcal{X}} \mathcal{F}(\mathcal{X})$$

where

$$\mathcal{F}(\mathcal{X}) = \sum_{r \in R} \left[1 - \prod_{x \in \mathcal{X}_r} (1 - x)\right].$$

Here, $\mathcal{X}_r$ is the set of variables such that $x = x(s)$ and $r(s) = r$.

COSP has been shown to be a challenging problem for DCOP methods. Typical COSP instances have hundreds of agents and thousands of requests, leading to millions of decision variables. In addition, the constraint graph of COSP has high degrees on the order of $\Omega(|A| \cdot |R|)$. The constraint graph is also assumed to be only locally known to an agent. An agent a is oblivious to all tasks $s \notin S_a$ and variables $x \notin \mathcal{X}_a$. This is not consistent with standard DCOPs in which agents know all neighboring variables/agents in the constraint graph [13]. In COSP, it is assumed that agents know the existence of all other agents but have no knowledge of the variables of other agents. These factors make existing DCOP approaches that rely on agents communicating with neighboring agents in the constraint graph both computationally challenging due to the high degrees in the graph and inapplicable since we cannot assume agents know which agents they share constraints with.

3.2 Dynamic Multi-Satellite Constellation Observation Scheduling

We now present the *Dynamic Multi-Satellite Constellation Observation Scheduling Problem* (DCOSP) and discuss how previous challenges from COSP transfer to DCOSP.

A DDCOP consists of a set of T sequential DCOPs. We define DCOSP similarly as a set of COSP instances. Let δ_t be the COSP instance at time t. We then define the DCOSP, δ, as $\delta = \{\delta_t\}_{t=0}^{T}$. We assume that the agents have no prior knowledge of when or how the problem might change and must act reactively. We refer to the requests and variables of δ_t as R^{δ_t} and $\mathcal{X}^{\delta_t}$. Note that δ_i depends on δ_j for $j < i$ since these prior DCOPs will determine the starting state of δ_i. For example, resource expenditure affects both current and future solutions. We assume that there is a globally known horizon for a DCOSP instance, $h(\delta) = [h_s(\delta), h_e(\delta)]$ and that the horizon of each COSP instance is $h(\delta_t) = [h_s(\delta_t), h_e(\delta)]$ where $h_s(\delta_t) \in h(\delta)$.

The utility of DCOSP is not the same as a typical DDCOP. In a typical DDCOP, the utility is defined as the sum of the utility functions of the individual DCOPs, which means that an optimal solution is obtained by solving each DCOP optimally in sequence. However, this formulation does not adequately capture DCOSP utility. We define the utility of DCOSP to be the number of requests that are satisfied, where satisfaction is determined by an observation for a request being *executed*. This utility rewards completing a task rather than just scheduling one. Due to the online nature of DCOSP, the scheduling horizon overlaps the execution horizon. Therefore, scheduling an observation does not guarantee that it will be executed. Consider that a task for request r is scheduled at time t_0 in δ_{t_0} to be executed at time t_i. If the task is then unscheduled at some δ_{t_j} where $0 < j < i$ then r will not be satisfied despite having a task scheduled in COSP instance δ_{t_0}.

To formally define this utility, we provide some useful definitions. Let $\bar{h}(\delta_t)$ be the unknown execution horizon of δ_t. We visually show $\bar{h}(\delta_t)$ in Figure 1. This is the horizon for which the problem is static and is defined by δ_t. Formally,

$$\bar{h}(\delta_t) = \begin{cases} [h_s(\delta_t), h_s(\delta_{t+1})] & \text{if } t < T \\ [h_s(\delta_t), h_e(\delta_t)] & \text{else (i.e. } t = T). \end{cases}$$

We then define the proposition $\text{executed}(x)$ as

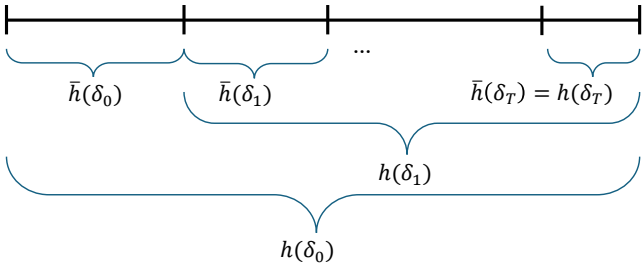


Figure 1: We illustrate $\bar{h}(\delta_t)$; the line spans the entire horizon of the DCOSP δ where $\bar{h}(\delta_t)$ shows the unknown interval the problem is static as defined by δ_t.

$$\text{executed}(x) = x \cdot \mathbb{I}[\exists t\ x = x(s) \wedge h(s) \cap \bar{h}(\delta_t) \neq \varnothing].$$

The above is equal to one if and only if $x = 1$, meaning task s was scheduled, and the horizon of s occurred during the time when the problem is static as defined by δ_t. This means that the task was scheduled when it was executed. Using this proposition, we can define the utility of DCOSP for an assignment of variables X^δ over all time steps.

$$\mathcal{F}(X^\delta) = \sum_{r \in R^\delta} \left[1 - \prod_{x \in X_r^\delta} (1 - \text{executed}(x)) \right]$$

where $R^\delta = \bigcup_{t=0}^{T} R^{\delta_t}$ and $X_r^\delta = \bigcup_{t=0}^{T} X_r^{\delta_t}$. R^δ is the set of all requests in DCOSP δ and X_r^δ are all tasks for request r over δ. This utility function rewards observations that are completed, not just scheduled.

The same challenges of solving COSP are multiplied when solving DCOSP. Solving a single instance of COSP is computationally challenging for most DCOP methods. Solutions that are linear in the maximum degree of the constraint graph suffer due to COSP instances having degrees $\Omega(|A| \cdot |R|)$. We again assume that the constraint graph is only partially known to an agent. Therefore, solutions to DCOSP need to conform to this assumption that cross-agent edges in the constraint graph are unknown and to be efficient in computation and communication.

4 ALGORITHMS

4.1 Obtaining an Optimal Solution to DCOSP

There is no clear online algorithm for computing an optimal DCOSP solution. This is due to agents having no prior knowledge of problem dynamics and optimally solving each individual COSP instance not necessarily constituting an optimal DCOSP solution. However, we can obtain an optimal solution by collapsing DCOSP into a single DCOP. This relies on using omniscient knowledge of the problem dynamics and is therefore not feasible in actuality.

We can collapse a DCOSP δ into a DCOP δ' to reason only about the observations that matter. The following are the key set constructions of δ':

$$S^{\delta'} = \{s \in S^\delta \mid \exists_t\ h(s) \cap \bar{h}(\delta_t) \neq \varnothing\} \text{ and}$$

$$X^{\delta'} = \{x \in X^\delta \mid \exists_t\ x = x(s) \wedge h(s) \cap \bar{h}(\delta_t) \neq \varnothing\}.$$

By construction, $\text{executed}(x) = x$ if and only if $x \in X^{\delta'}$ and $\text{executed}(x) = 0$ otherwise. Therefore, we obtain the following equivalence.

$$\begin{aligned}\mathcal{F}(X^\delta) &= \sum_{r \in R^\delta} \left[1 - \prod_{x \in X_r^\delta} (1 - \text{executed}(x)) \right] \\ &= \sum_{r \in R^\delta} \left[1 - \prod_{x \in X_r^{\delta'}} (1 - x) \right] \\ &= \mathcal{F}(X^{\delta'}).\end{aligned}$$

Solving δ' optimally results in an optimal solution to the DCOSP δ. By definition, δ' is a static COSP instance. Therefore, we can employ any complete algorithm to obtain an optimal solution to δ by solving δ'. Although the omniscient solver is not deployable, it serves as a key performance upper bound for benchmarking practical, online algorithms such as D-NSS. However, even solving a single DCOP is NP-Hard [29]. Therefore, for large DCOSP instances, we use an incomplete solver to obtain a near-optimal solution since any exponential time search is infeasible.

4.2 Dynamic Incremental Neighborhood Stochastic Search

In this section, we present the *Dynamic Incremental Neighborhood Stochastic Search* algorithm (D-NSS) shown in Algorithm 1. Due to the scale of DCOSP and the computational constraints of satellites, we require DDCOP algorithms that are both efficient and can reason about the dynamic nature of the problem. D-NSS extends NSS to address the drawbacks of general DDCOP algorithms. NSS is an iterative algorithm where at each iteration agents stochastically update their variable assignments based on the assignments of agents they communicate with. This iterative procedure is shared with other algorithms such as DSA. However, NSS relies on a decomposition heuristic $\Upsilon : A \times S \to \{0, 1\}$ to generate a subproblem $\mathcal{N}$ for neighborhoods of agents to solve. This subproblem is a smaller DCOP consisting of requests $R_\mathcal{N}$ and agents $A_\mathcal{N}$. D-NSS continually computes subproblems, repairs local solutions between problem instances, and reasons about prior scheduling and execution in the search and repair phases.

We leverage the Geometric Neighborhood Decomposition heuristic (GND) to create these subproblems every time dynamics occur [53]. GND efficiently allocates requests to neighborhoods of agents based on the orbital geometry of the constellation and has been shown to effectively partition COSP instances. For completeness,

Algorithm 1 Dynamic Incremental Neighborhood Stochastic Search (D-NSS) for agent a

Input: $sched, R, A, S_a, C_a, \Upsilon, maxIters$
Output: Schedule for agent a

1: $\mathcal{N} =$ computeSubProblem(a, A, R, S_a, Υ)
2: $sched =$ repair$(sched, R_\mathcal{N}, S_a, C_a)$
3: $sched =$ D-NSS-Search$(sched, R_\mathcal{N}, A_\mathcal{N}, S_a, C_a, maxIters)$
4: **return** $sched$

Algorithm 2 REPAIR for agent a

Input: $sched, R, S_a, C_a$
Output: Repaired Schedule for agent a

```
for s ∈ sched do
    if r(s) ∉ R then
        removeFromSchedule(sched, s)
shuffle S_a
for s ∈ S_a do
    if s satisfies C_a ∧ r(s) ∉ sched then
        addToSchedule(sched, s)
return sched
```

Algorithm 3 D-NSS-SEARCH for agent a

Input: $sched, R_{\mathcal{N}}, A_{\mathcal{N}}, S_a, C_a, maxIters$
Output: Schedule for agent a

```
while i < maxIters ∧ not converged do
    com_out, com_in = message(A_N, sched)
    shuffle R_N
    for r ∈ R_N do
        assigned = stochasticUpdate(r, sched, com_in)
        if assigned = true then
            sched = schedule(r, sched, S_a)
    i ← i + 1
return sched
```

we provide an outline of GND. GND was first introduced with the NSS algorithm and we refer the reader to prior work for a full presentation of the heuristic [53].

GND leverages the orbital geometry of the satellite constellation to hierarchically partition requests to neighborhoods of agents. Let K be the set of orbital planes that define a satellite constellation[1]. For every request, an agent computes the supply from all orbital planes and adds these to estimate the total number of agents with overflights for the request. The supply can also be thought of as an estimate of degrees in the constraint graph. Iterating through requests in ascending order of supply, a request gets assigned to the n neighborhoods with the highest ratio of supply to temporal conflicts. Temporal conflicts are counts of other requests already allocated to a neighborhood that overlap in time with a given request. Finally, within a neighborhood, requests are further subdivided to agents based on biases towards specific tiles on Earth. GND with n degrees of incompleteness is denoted GND(n).

D-NSS restarts the search phase when changes are initiated in the problem. A key procedure is the REPAIR function that repairs previously computed solutions to leverage assignments of unchanging variables shown in Algorithm 2. Repairing consists of removing all tasks that are no longer in the current problem instance and greedily inserting new tasks in random order. Note that we can also remove from an agent's schedule all requests that have been previously executed by agents in the same neighborhood. One benefit of the D-NSS algorithm is that it can leverage different repair procedures. We use the random repair function for two main reasons. Random initialization is the standard for DSA and NSS variants in static domains [49] as it promotes diverse solutions and D-NSS is designed to be as lightweight as possible. Computation-intensive repair procedures counteract the efficiency of the algorithm. We show later on that performing random repair improves the quality and efficiency of D-NSS on DCOSP instances.

After each agent repairs its solution, all agents synchronously begin the iteration phase, D-NSS-SEARCH, to fine-tune the repaired solutions. D-NSS-SEARCH extends the search phase of NSS to account for tasks that have just been scheduled versus ones that have been executed. We detail the following sub-procedures.

- computeSubProblem(a, A, R, S_a, Υ). This function computes the neighborhood of agent a and the subset of requests for that neighborhood using the decomposition heuristic Υ. We use Υ = GND. GND produces neighborhoods that are reflexive and transitive.
- message$(A_{\mathcal{N}}, sched)$. This function defines the message exchange between agents in a neighborhood. Each agent a sends to each other agent in $A_{\mathcal{N}} \backslash \{a\}$ the subset of $R_{\mathcal{N}}$ that it has executed a task for already and the subset that it has scheduled in the previous iteration via the variable com_out. The resulting data structure com_in contains the satisfaction information for the neighborhood.
- stochasticUpdate$(r, sched, com_in)$. This function computes the assignment of an agent and a request based on the neighborhood's communication. An assignment refers to if an agent should attempt to schedule a specific request. Let W be the count of agents that scheduled or executed r in the previous iteration, the probability P_u be a hyperparamater, executed(r) be the predicate that r was executed already, and assigned(a, r) be the predicate that a is assigned to r. An agent computes the probability of assigning to r in the next iteration, $P(a, r|com\_in)$ using the update scheme from Table 1. For example, according to com_in, if request r has not been executed, agent a is not assigned to it, and $W \geqslant 1$, agent a will always remain unassigned to r. This update scheme extends the static NSS update schemes to account for dynamic scheduling and execution. If a task for a request has already been executed, then all agents should unassign. Note that executed$(r) \implies W \geqslant 1$.
- schedule$(r, sched, S_a)$. This function tries to schedule a task for request r given the current schedule. If a task for r satisfies C_a it is inserted into the schedule. Otherwise, the scheduler may remove a single task from the schedule to satisfy the constraints. The task with the closest start time to the new task is used as a heuristic for removal. Agent a remains

	executed(r)	$\neg$executed(r) $\neg$assigned(a, r)	$\neg$executed(r) assigned(a, r)
$W = 0$	N/A	1	$1 - P_u$
$W \geqslant 1$	0	0	$1/W$

Table 1: Stochastic assignment update scheme for agent a. The table values denote the probability that agent a assigns to request r based on com_in, $P(a, r|com\_in)$.

[1]An orbital plane is a fixed orbit around Earth that many satellites may follow at various spacing.

assigned to a removed task and can attempt to re-schedule it in subsequent iterations. Task removal enables the search phase to overcome local maxima. Note that vanilla DCOSP considers all requests of equal priority and there is no temporal flexibility in the start or end times of tasks. However, DCOSP and the scheduling procedure are amenable to these extensions.

Without prior knowledge of the problem dynamics, it is difficult for any online algorithm to reason about which scheduled tasks will be executed. Prioritizing requests that are earlier in the horizon may be more likely not to be removed and yield reward. However, expending resources early on in the horizon results in fewer resources available to handle problem dynamics later. Although D-NSS currently focuses on reactive repair, it can be extended to integrate proactive scheduling through predictive models, which is a subject of future work.

We introduce variables to analyze the complexity of D-NSS. Let L be the maximum size of a satellite's schedule, $A_{\mathcal{N}}$ be the largest set of agents in a sub-problem, and $R_{\mathcal{N}}$ be the largest set of requests in a sub-problem. In general $L \ll |R_{\mathcal{N}}| \ll |R|$ since L is constrained by resources and time. Sub-problem computation via GND has a time complexity of $O(|R_{\mathcal{N}}|)$ and uses no communication. The REPAIR procedure is individually computed by an agent and takes $O(|R_{\mathcal{N}}| \log L)$ time and again uses no communication. D-NSS inherits from NSS a computation and communication complexity of $O(|A_{\mathcal{N}}| \cdot |R_{\mathcal{N}}|)$ during an iteration.

4.3 Baseline Algorithms

We evaluate several baseline algorithms in addition to D-NSS. The naive alternative to D-NSS is to recompute a solution from scratch every time problem dynamics occur. We refer to this procedure as 0-NSS. 0-NSS has the same theoretical complexities as D-NSS. We also evaluate dynamic variations of the *Distributed Stochastic Search Algorithm* (DSA) [33, 49]. D-DSA uses Algorithm 2 to repair DSA solutions. Likewise, 0-DSA runs DSA from scratch every time the problem changes. These DSA variants have a complexity of $O(|A| \cdot |R|)$ per iteration. Dynamic DSA is one of the most lightweight DDCOP solvers and can be deployed without violating the constraint graph assumptions of DCOSP. Other DDCOP algorithms such as a dynamic variation of MGM [26], would also incur a computation and communication complexity of $O(|A| \cdot |R|)$ per iteration. Due to the scale of the problem instances and the constraints of satellite computation and communication, we are unable to evaluate any exponential-time algorithms.

We include two baseline non-communication-reliant algorithms: random and greedy. Both of these algorithms construct a schedule for a satellite without reasoning about other agents. These algorithms transition schedules after the dynamics occur by removing redundant tasks. The random solver randomly inserts tasks into an agent's schedule while the greedy algorithm orders tasks by their start time and iteratively inserts them into a schedule in a single pass. Although simple, a greedy solver is comparable to deployed planners on spacecraft [15].

Finally, we use the construction outlined previously to obtain optimal and near-optimal solutions. For small problems, we obtain an optimal solution by constructing a static COSP instance and solving it with a centralized branch and bound. For large problems, computing an optimal solution is not computationally feasible, so we lower bound the optimal solution using *Squeaky Wheel Optimization* [21], an incomplete centralized solver.

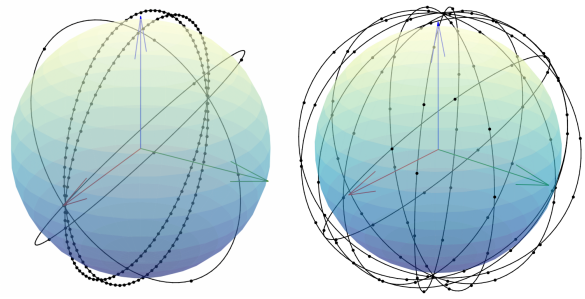


Figure 2: The satellite constellations: Planet (left) and Walker (right). Dots show satellites in an orbital plane.

5 EXPERIMENTS

5.1 Setup

In this section, we outline the experimental setup including constellations and dynamic observation campaigns.

5.1.1 Satellite Constellations. We evaluate two constellations modeled after operational low Earth orbit constellations [38]. The Planet constellation is modeled on the Dove constellation from Planet Labs. This constellation is composed of two near sun-synchronous orbital planes at 95° inclinations each composed of 95 satellites with an additional two orbital planes at 52° inclinations each with 5 satellites. The Walker constellation is motivated by the Skysat constellation from Planet Labs. This constellation has 6 orbital planes with 14 satellites each at an 88° inclination and an overlay of 2 orbital planes at a 51.6° inclination with 12 satellites. Each satellite has a memory capacity of 125 GB and a single sensor that can slew to 60° and 45° off-nadir for the Planet and Walker constellations respectively. Figure 2 depicts these two constellations.

5.1.2 Downlinks. A 62.5 MB/s constant bit stream models the satellite downlink during visibility periods with a ground station. We incorporate two ground stations: the ASF Near Space Network Satellite Tracking Ground Station and the Guam Remote Ground Terminal System.

5.1.3 Dynamic Observation Request Campaigns. The target set, $\mathcal{T}$, is 634 globally distributed cities. Dynamic campaigns consist of periodic requests of these targets. For large problem instances, a periodicity is uniformly sampled from the range $[5, 12]$. A target with periodicity p is requested to be observed once within p evenly spaced intervals. For small problem instances, we fix the periodicity at 3. The scheduling horizon is set to be one day. However, the start of the horizon is randomly initialized. An observation's memory consumption is sampled from a normal distribution with mean 50 MB and standard deviation 10 MB. The time interval required to execute an observation is 63 seconds which accounts for imaging, slewing, and processing.

To generate dynamics, we select a volatility parameter v. This determines the number of changes during the horizon. The time of the dynamics is uniformly distributed over the final $1 - \frac{2}{3 \cdot v}$ of the scheduling horizon. This ensures that dynamics do not occur directly after the start of the horizon. The set of starting requests is randomly selected from the request set and initialized to one-third of the total size. We select from the remaining requests a random set of $\frac{2}{3 \cdot v}$ to add. From the active requests, we randomly remove $\frac{1}{3 \cdot v}$ of them. We enforce that once a request is removed, it is not added back, and requests are not changed after their execution horizon starts. We sample $v \in [3, 5]$ uniformly.

5.1.4 Hyperparameters and Execution Environment. The stochastic update of NSS and DSA relies on the probability, P_u that determines when an agent should unassign from a task. We fix this value at 0.7, as published in previous work [33, 53]. The *maxIters* is set to 20. For the GND heuristic, we set all hyperparameters as described in previous work, and specifically use the GND(2) heuristic [53]. We use a random seed of 2005 for the initial scenario generation. When evaluating N scenarios, we increment this seed for generation of each subsequent simulation. The random seed used in the REPAIR procedure is initialized to 1. The NSS and DSA algorithms are seeded with a value of 1234, and the random seed for GND is set to 2. The random scheduling algorithm uses a seed of 2023. All experiments are executed using Java 19 on a MacBook Pro 16 laptop with an M2 Max processor (12-core CPU and 38-core GPU) and 64 GB of RAM.

5.2 Results on Small Problem Instances

We are able to obtain an optimal solution for small problem instances using the omniscient offline algorithm. We solve this DCOP with a centralized branch and bound to obtain an optimal schedule for each satellite. For the Planet constellation, we solve campaigns of up to 500 requests. For the Walker constellation, this increases to 1000 requests. In addition to having fewer agents, the Walker constellation geometry under-constrains small problems, making finding optimal solutions faster. An optimal solver does not consistently terminate for larger problems. We report the average gap in satisfaction percentage to the optimal solution for 10 dynamic small problem instances in Tables 2 and 3.

Algorithm	Opt. Gap (%)	Time (ms)	Messages (KB)
Random	2.530	<1	0
Greedy	8.373	<1	0
D-NSS	**1.867**	**<1**	**7.3**
0-NSS	2.590	<1	13.2
D-DSA	4.217	5.3	980.6
0-DSA	3.795	4.7	718.4

Table 2: Results on 10 dynamic problems for the Planet constellation (up to 500 requests).

Algorithm	Opt. Gap (%)	Time (ms)	Messages (KB)
Random	14.945	<1	0
Greedy	15.604	<1	0
D-NSS	**0.142**	**5.2**	**240.2**
0-NSS	0.480	6.1	400.0
D-DSA	1.165	58.2	10,459.5
0-DSA	1.215	54.0	7,789.0

Table 3: Results on 10 dynamic problems for the Walker constellation (up to 1000 requests).

These results support the theoretical analysis of D-NSS and demonstrate near-optimal performance. D-NSS outperforms all baselines, including D-DSA, 0-DSA, and 0-NSS, and does so while using less computation and communication. Notably, compared to DSA variants, D-NSS finds better solutions using an order of magnitude less computation and up to two orders of magnitude less communication. This is due to both the lower theoretical complexity per iteration and the faster convergence of D-NSS.

5.3 Results on Large Problem Instances

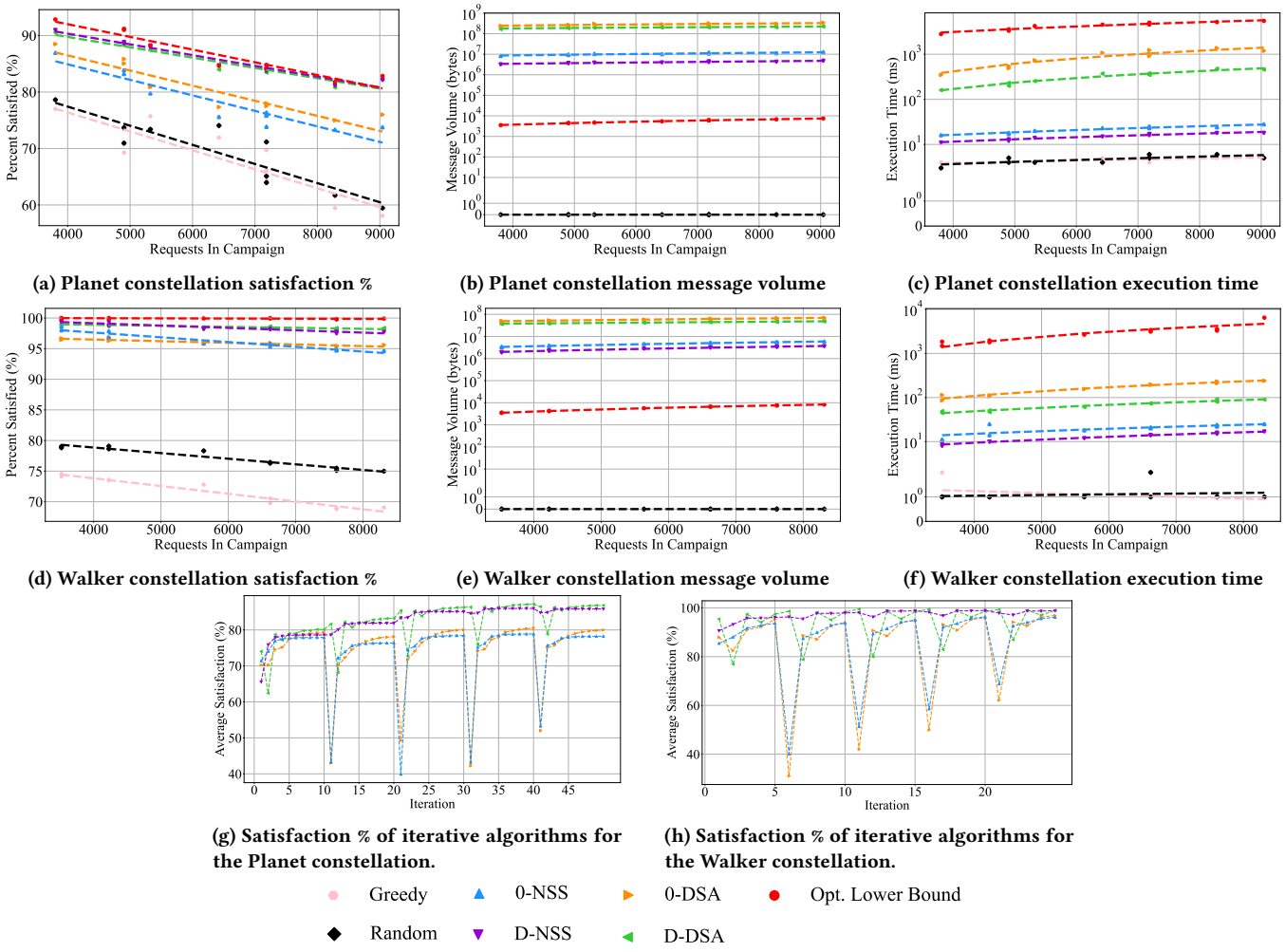
We evaluate the algorithms on realistic, large-scale, dynamic scenarios with thousands of requests. Figure 3 shows the results for the Planet and Walker constellations. We report the total utility of each algorithm in Figures 3a and 3d, the total message volume in Figures 3b and 3e, and the average per-agent execution time in Figures 3c and 3f. Note the log scale in the figures for message volume and runtime.

In terms of solution quality, D-NSS and D-DSA achieve close to the optimal lower-bound solution. D-NSS outperforms both 0-NSS and 0-DSA as well as the greedy and random baselines. Crucially, D-NSS achieves high solution quality while having a lower total message volume and execution time compared to D-DSA, 0-DSA, and 0-NSS. Both D-NSS and 0-NSS use an order of magnitude fewer messages and computation times than their DSA counterparts. Since the optimal lower bound algorithm is computed centrally, the message volume corresponds to the schedules a ground station would have to uplink.

We also evaluate the stability and convergence of D-NSS compared to D-DSA, 0-DSA, and 0-NSS. Figures 3g and 3h show the average solution quality over iterations of the algorithms during large-problem instances with $v = 5$. We fix the number of iterations for all algorithms to show the relative performance. Clearly, D-NSS and D-DSA are much more stable, reacting much less volatility to problem dynamics, illustrated by the sharp drops in solution quality of 0-NSS and 0-DSA. Notably, D-NSS is very stable; even in the iteration directly after problem dynamics, D-NSS repairs solutions effectively to maintain or improve solution quality. This supports computing and repairing sub-problems rather than global solutions like D-DSA. In addition, the stability of D-NSS leads to a quick convergence in practice, which drives the efficiency of the algorithm.

6 CONCLUSION

Large problems remain a challenge for DDCOP algorithms due to their computational and communication complexities. Even linear time and messaging solutions may not be feasible when constraint graphs are highly connected or agents have limited compute. These challenges apply to DCOSP. We present the D-NSS algorithm, a decomposition-based algorithm that is efficient in time and message



(a) Planet constellation satisfaction % **(b) Planet constellation message volume** **(c) Planet constellation execution time**

(d) Walker constellation satisfaction % **(e) Walker constellation message volume** **(f) Walker constellation execution time**

(g) Satisfaction % of iterative algorithms for the Planet constellation. **(h) Satisfaction % of iterative algorithms for the Walker constellation.**

Figure 3: Results of 10 large-scale simulations for the Planet (top row) and Walker (middle row) constellation. We report the average satisfaction, total message volume, and per-agent runtime. Note the log scale for message volume and runtime. We also show the solution quality across a fixed number of iterations for iterative algorithms for instances with $v = 5$ (bottom).

complexity and can scale to problems much larger than previous approaches. We show that D-NSS stabilizes quickly and uses an order of magnitude fewer messages and compute times compared to DDCOP baselines. Despite no quality guarantees, we show that D-NSS computes near-optimal solutions.

D-NSS generalizes to many DDCOP problems. While we leverage the GND heuristic, the D-NSS framework can accommodate alternative heuristics, including learned or adaptive ones, enabling broader applicability across DDCOP domains. Some examples include multi-agent path finding [42], mobile sensor teams [34], and UAV coordination [39].

In future work, we can extend D-NSS to reason about future dynamics. We can construct or learn heuristics that prioritize tasks that are more likely to yield utility when executed. Alternatively, we can extend DCOSP to be a *Proactive DCOP* (PDDCOP) [20] where we have priors on problem dynamics, and agents can generate robust offline solutions. However, PDDCOPs require a model of problem dynamics which may not be obtainable in practice.

DCOSP solutions will be demonstrated in operational scenarios beginning in 2026. One example is NASA's FAME demonstration [9] which centers around a *federated* observation system [50]. The FAME demonstration consists of more than 60 participating Earth-observing spacecraft that will coordinate their measurements to optimize observation completion across dynamic scenarios. Through onboard control, satellites will be able to react faster and with more precision to time-sensitive events such as natural disasters. These observations will provide novel measurements of scientific processes and crucial, up-to-date information for human operators. This work has laid the foundation for dynamic, large-scale, distributed onboard scheduling of Earth-observing satellites.

ACKNOWLEDGMENTS

Portions of the research were carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). Government sponsorship acknowledged.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE2140739. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Sean Augenstein, Alejandra Estanislao, Emmanuel Guere, and Sean Blaes. 2016. 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 and Scheduling*, Vol. 26. 345–352.

[2] Graham Billiau, Chee Fon Chang, and Aditya Ghose. 2010. SBDO: A new robust approach to dynamic distributed constraint optimisation. In *International Conference on Principles and Practice of Multi-Agent Systems*. 11–26.

[3] James Boerkoel, James Mason, Daniel Wang, Steve Chien, and Adrien Maillard. 2021. An efficient approach for scheduling imaging tasks across a fleet of satellites. In *Proceedings of the International Workshop on Planning and Scheduling for Space*.

[4] Grégory Bonnet and Catherine Tessier. 2007. Collaboration among a satellite swarm. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 1–8.

[5] Grégory Bonnet and Catherine Tessier. 2008. Coordination despite constrained communications: A satellite constellation case. In *Proceedings of the National Conference on Control Architectures of Robots*. 89–100.

[6] Abhijit Chatterjee and Ratnasingham Tharmarasa. 2024. Multi-stage optimization framework of satellite scheduling for large areas of interest. *Advances in Space Research* 73, 3 (2024).

[7] Steve Chien, Alberto Candela, Itai Zilberstein, David Rijlaarsdam, Tom Hendrix, and Aubrey Dunne. 2024. Leveraging commerical assets, edge computing, and near real-time communications for an enhanced New Observing Strategies (NOS) flight demonstration. In *Proceedings of the IEEE Geoscience and Remote Sensing Symposium*.

[8] Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castano, Ashley Davis, Dan Mandl, Stuart Frye, Bruce Trout, et al. 2005. Using autonomy flight software to improve science return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication* 2, 4 (2005), 196–216.

[9] Steve Chien, Itai Zilberstein, Alberto Candela, Domenico Barretta, David Rijlaarsdam, Tom Hendrix, Aubrey Dunne, Oriol Cortés Grauc, Alexandre Gol i Mestrec, Manel Pedra Bovec, Oriol Aragon, Juan Puig Miquel, Arvind Subramanian, Vishesh Vatsal, Adithya Kothandhapani, Jad Mogannam, and Mitchell Scher. 2025. Multi-asset New Observing Systems flight demonstration. In *Proceedings of the International Conference on Space Operations*.

[10] Steve Chien, Itai Zilberstein, Alberto Candela, David Rijlaarsdam, Amaury Perrocheau, Aubrey Dunne, Tom Hendrix, Oriol Cortés Grauc, Alexandre Gol i Mestrec, Manel Pedra Bovec, Oriol Aragon, and Juan Puig Miquel. 2025. Flight of dynamic targeting on CogniSAT-6 - Update. In *Proceedings of the International Conference on Space Operations*.

[11] Duncan Eddy and Mykel J Kochenderfer. 2021. A maximum independent set method for scheduling Earth-observing satellite constellations. *Journal of Spacecraft and Rockets* 58, 5 (2021), 1416–1429.

[12] Adrian Petcu Boi Faltings. 2005. Superstabilizing, fault-containing multiagent combinatorial optimization. In *AAAI*. 449–454.

[13] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research* 61 (2018), 623–698.

[14] Ferdinando Fioretto, William Yeoh, Enrico Pontelli, Ye Ma, and Satishkumar J Ranade. 2017. A distributed constraint optimization (DCOP) approach to the economic dispatch with demand response. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 999–1007.

[15] Dan Gaines, Steve Chien, Gregg Rabideau, Stephen Kuhn, Vincent Wong, Amruta Yelamanchili, Shannon Towey, Jagriti Agrawal, Wayne Chi, Andrea Connell, Evan Davis, and Colette Lohr. 2022. Onboard planning for the Mars 2020 Perseverance Rover. In *Symposium on Advanced Space Technologies in Robotics and Automation*.

[16] Amir Gershman, Amnon Meisels, and Roie Zivan. 2009. Asynchronous forward bounding for distributed COPs. *Journal of Artificial Intelligence Research* 34 (2009), 61–88.

[17] Al Globus, James Crawford, Jason Lohn, and Anna Pryor. 2004. A comparison of techniques for scheduling Earth-observing satellites. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*.

[18] Lei He, Xiaolu Liu, Gilbert Laporte, Yingwu Chen, and Yingguo Chen. 2018. An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. *Computers & Operations Research* 100 (2018), 12–25.

[19] Katsutoshi Hirayama and Makoto Yokoo. 1997. Distributed partial constraint satisfaction problem. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*. 222–236.

[20] Khoi D Hoang, Ferdinando Fioretto, Ping Hou, William Yeoh, Makoto Yokoo, and Roie Zivan. 2022. Proactive dynamic distributed constraint optimization problems. *Journal of Artificial Intelligence Research* 74 (2022), 179–225.

[21] David E Joslin and David P Clements. 1999. Squeaky wheel optimization. *Journal of Artificial Intelligence Research* 10 (1999), 353–373.

[22] Sankalp Khanna, Abdul Sattar, David Hansen, and Bela Stantic. 2009. An efficient algorithm for solving dynamic complex DCOP problems. In *International Joint Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 2. 339–346.

[23] Parimal Kopardekar and Laurie Grindle. 2021. NASA ARMD Wildfire Management Workshop. (2021).

[24] Robert N Lass, Evan Sultanik, and William C Regli. 2008. Dynamic distributed constraint reasoning. In *AAAI*. 1466–1469.

[25] Xiang Lin, Yuning Chen, Junhua Xue, Boquan Zhang, Lei He, and Yingwu Chen. 2024. Large-volume LEO satellite imaging data networked transmission scheduling problem: Model and algorithm. *Expert Systems with Applications* 249 (2024), 123649.

[26] Rajiv T Maheswaran, Jonathan P Pearce, and Milind Tambe. 2004. Distributed algorithms for DCOP: A graphical-game-based approach. In *Proceedings of the International Conference on Parallel and Distributed Computing Systems*. 432–439.

[27] Roger Mailler. 2005. Comparing two approaches to dynamic, distributed constraint satisfaction. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*. 1049–1056.

[28] Roger Mailler and Victor Lesser. 2004. Solving distributed constraint optimization problems using cooperative mediation. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 438–445.

[29] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. 2005. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence* 161, 1-2 (2005), 149–180.

[30] Sreeja Nag, Alan S Li, and James H Merrick. 2018. Scheduling algorithms for rapid imaging using agile Cubesat constellations. *Advances in Space Research* 61, 3 (2018), 891–913.

[31] NewSpace. 2023. NewSpace Constellations. https://www.newspace.im. Accessed: 2025-05-01.

[32] Duc Thien Nguyen, William Yeoh, Hoong Chuin Lau, and Roie Zivan. 2019. Distributed Gibbs: A linear-space sampling-based DCOP algorithm. *Journal of Artificial Intelligence Research* 64 (2019), 705–748.

[33] Shreya Parjan and Steve A. Chien. 2023. Decentralized observation allocation for a large-scale constellation. *Journal of Aerospace Information Systems* (2023), 1–15.

[34] Arseniy Pertzovsky, Roie Zivan, and Noa Agmon. 2024. Collision avoiding Max-Sum for mobile sensor teams. *Journal of Artificial Intelligence Research* 79 (2024), 1281–1311.

[35] Adrian Petcu and Boi Faltings. 2005. DPOP: A scalable method for multiagent constraint optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 266–271.

[36] Sean Phillips and Fernando Parra. 2021. A case study on auction-based task allocation algorithms in multi-satellite systems. In *Proceedings of AIAA Scitech*.

[37] Gauthier Picard. 2022. Auction-based and distributed optimization approaches for scheduling observations in satellite constellations with exclusive orbit portions. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 1056–2064.

[38] Planet. 2023. Our Constellations. https://www.planet.com/our-constellations. Accessed: 2025-05-01.

[39] Marc Pujol-Gonzalez, Jesus Cerquides, Pedro Meseguer, Juan Antonio Rodríguez-Aguilar, and Milind Tambe. 2013. Engineering the decentralized coordination of UAVs with limited communication range. *Advances in Artificial Intelligence* 1 (2013), 199–208.

[40] Gregg Rabideau, Joseph Russino, Andrew Branch, Nihal Dhamani, Tiago Stegun Vaquero, Steve Chien, Jean-Pierre de la Croix, and Federico Rossi. 2025. Planning, scheduling, and execution on the Moon: the CADRE technology demonstration mission. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*.

[41] Anton Ridgway and Roger Mailler. 2015. Dynamic theoretical analysis of the distributed stochastic and distributed breakout algorithms. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 405–412.

[42] Oren Salzman and Roni Stern. 2020. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. 1711–1715.

[43] Vishwa Shah, Vivek Vittaldev, Leon Stepan, and Cyrus Foster. 2019. Scheduling the world's largest Earth-observing fleet of medium-resolution imaging satellites. In *Proceedings of the International Workshop on Planning and Scheduling for Space*. 156–161.

[44] Samuel Squillaci, Cédric Pralet, and Stéphanie Roussel. 2023. Scheduling complex observation requests for a constellation of satellites: Large neighborhood search approaches. In *Proceedings of the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. 443–459.

[45] Samuel Squillaci, Stéphanie Roussel, and Cédric Pralet. 2021. Managing complex requests for a constellation of Earth-observing satellites. In *Proceedings of the International Workshop on Planning and Scheduling for Space*.

[46] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nick Jennings. 2009. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 299––304.

[47] Xinwei Wang, Guohua Wu, Lining Xing, and Witold Pedrycz. 2020. Agile Earth observation satellite scheduling over 20 years: Formulations, methods, and future directions. *IEEE Systems Journal* 15, 3 (2020), 3881–3892.

[48] William Yeoh, Pradeep Varakantham, Xiaoxun Sun, and Sven Koenig. 2015. Incremental DCOP search algorithms for solving dynamic DCOP problems. In *International Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 2. 257–264.

[49] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. 2005. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161, 1-2 (2005), 55–87.

[50] Itai Zilberstein, Alberto Candela, and Steve Chien. 2025. Federated autonomous operations: A New paradigm for large-scale observation systems. In *Proceedings of the International Conference on Space Operations*.

[51] Itai Zilberstein, Alberto Candela, Steve Chien, David Rijlaarsdam, Tom Hendrix, Léonie Buckley, and Aubrey Dunne. 2024. Demonstrating onboard inference for Earth science applications with spectral analysis algorithms and deep learning. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

[52] Itai Zilberstein and Steve Chien. 2026. Large-Scale Continual Scheduling and Execution for Dynamic Distributed Satellite Constellation Observation Allocation. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*. Extended Abstract.

[53] Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. 2025. Decentralized, decomposition-based observation scheduling for a large-scale satellite constellation. *Journal of Artificial Intelligence Research* 82 (2025), 169–208.

[54] Roie Zivan, Harel Yedidsion, Steven Okamoto, Robin Glinton, and Katia Sycara. 2015. Distributed constraint optimization for teams of mobile sensing agents. *Autonomous Agents and Multiagent Systems* 29 (2015), 495–536.