DOI: 10.13009/EUCASS2025-248
11TH EUROPEAN CONFERENCE FOR AERONAUTICS AND SPACE SCIENCES (EUCASS)

Autonomous Decision-Making for Large Satellite Constellations: a Multi-Agent Reinforcement Learning Approach to Space Situational Awareness in Partially Observable Dynamic Environments

Clemente J. Juan Oliver[⋆†], *Vincenzo Messina*[⋆], *Sydney Dolan*[⋆] *and Alessandro Golkar*[⋆]
[⋆]*Technical University of Munich*
Caroline Herschel Strasse 100/II 85521 Ottobrunn - Germany
{clemente.juan, vincenzo.messina, sydney.dolan, golkar}@tum.de
[†]Corresponding author

Abstract

Traditional centralized operations struggle to scale with the complexity of large satellite constellations. This work presents a decentralized multi-agent deep reinforcement learning approach for operating an autonomous Space Situational Awareness constellation of 20 satellites. Four reward functions, with different reward scaling factors, are evaluated using a mission goal term across three coordination topologies. Developed strategies obtain similar or even outperform centralized rule-based and Mixed Integer Programming benchmarks in tested scenarios, and generalize across coordination topologies despite being trained in fully decentralized environments. Finally, space hardware inference confirms feasibility in modern systems, with execution times in the microsecond range.

1. Introduction

Managing large constellations is an increasingly challenging problem, due to the inherent complexity of space systems, the growth in constellation size - with up to hundreds or thousands of participants -, and the rapidly growing amount of objects and space debris in low Earth orbit (LEO)[1] . As a group of satellites working together as a system, a constellation needs to be managed in a way that ensures all of its participants have their tasks correctly scheduled and that they perform them properly. Spacecraft operators are responsible for this, as well as making sure that participants do not have faults or resource problems, while preventing them from colliding with other objects or debris. Due to these complex responsibilities, even in highly automated constellations like OneWeb and Starlink, satellite and flight dynamics operators frequently face high workloads when managing a large number of assets[2,3] . This highlights the need for further techniques and methods to improve the coordination and management of large, distributed systems.
Task scheduling of such systems is very complex due to the number of objects in orbit and the increasing sizes of constellations. There have been several proposals to address this issue using automated methods. These can be split into two main categories, namely ground and onboard automation. Offline building of the tasks using methods like Mixed Integer Programming (MIP), Ruled-Based or Machine Learning (ML) and then uplinking them through ground stations is a common approach to ground automation[4] , while the on-board approach, which is the chosen one in this work, gives satellites themselves decision-making capabilities to be able to evaluate their surrounding environment and current state to perform their task scheduling[5] . Enhancing autonomy onboard enables these types of spacecraft to take actions without the need to wait for ground commands, allowing quicker reactions to the changing environment of Earth's orbit. By allowing each satellite to choose between different operational modes autonomously, taking available onboard information into account, task scheduling can be planned and executed in real-time.
Task and spacecraft planning of resource-constrained multi-satellites has traditionally been a centralized, ground-based process. Spacecraft controllers have to prepare single schedules that dictate the actions of every satellite. This approach is reaching its limits when handling hundreds of satellites, when the use of exact methods or combinatorial optimization algorithms degrades the outcome of the missions[6–8] .
This decentralized model of information distribution, where each satellite has partial and limited information about its surrounding environment, can be modeled as a multi-agent reinforcement learning (MARL) scenario. MARL is based

DOI: 10.13009/EUCASS2025-248

on Reinforcement Learning (RL), where agents (the satellites in our case) try actions in an environment and receive rewards based on outcomes, learning optimal decision policies by trying to maximize these rewards over time[9] . The training process is computationally heavy and complex due to the need to simulate the environment, the interactions between satellites and update the policies themselves; however, the produced policies are relatively lightweight to execute. Therefore, these policies can be inferred in resource-constrained systems, such as constellations of CubeSats, running in a closed loop with few computational resources.
In the context of Space Situational Awareness (SSA), this work focuses on Space Surveillance and Tracking (SST), since opportunistic detection in these environments can greatly benefit from a more reactive and responsive approach to task scheduling. We formulate the constellation management problem as a group of observer small satellites, acting as agents, equipped with imaging and inter-satellite link (ISL) capabilities. Their main objective is to take images of other LEO orbiting target satellites and spread the information among them efficiently and collaboratively to enhance SSA operations. This is modeled using a decentralized partially observable Markov Decision Process (Dec-POMDP) framework, which has been widely studied in the area of MARL to formulate environments[9] .
The objective of this work is the development of a method that will allow large numbers of agents in LEO constellations to interact and complete a set of SSA mission tasks in an efficient manner with the use of MARL. We analyze the influence of different reward function terms and scaling factors in order to achieve better mission results. The main contributions of this paper are as follows:

- We formulate the satellite tasking problem as a collaborative Dec-POMDP, where each agent is equipped with inter-satellite links, power, storage, and specific imaging capabilities, and can choose between three operating modes (idle, communicate, observe) depending on their current state.
- We explore how to choose the scale and components of the reward function, thus improving the effectiveness of reward shaping in space missions.
- We evaluate the extensibility of our formulation across different levels of coordination topologies.

The structure of this paper is organized as follows. In Section 2, a brief overview of related studies is presented alongside their importance. Section 3 provides a theoretical background to the RL methodology followed and the simulation and reward function setup. Then, Section 4 describes our simulation results and provides insights on them. Lastly, conclusions and suggestions for further improvements are presented in Section 5.

2. Related Work

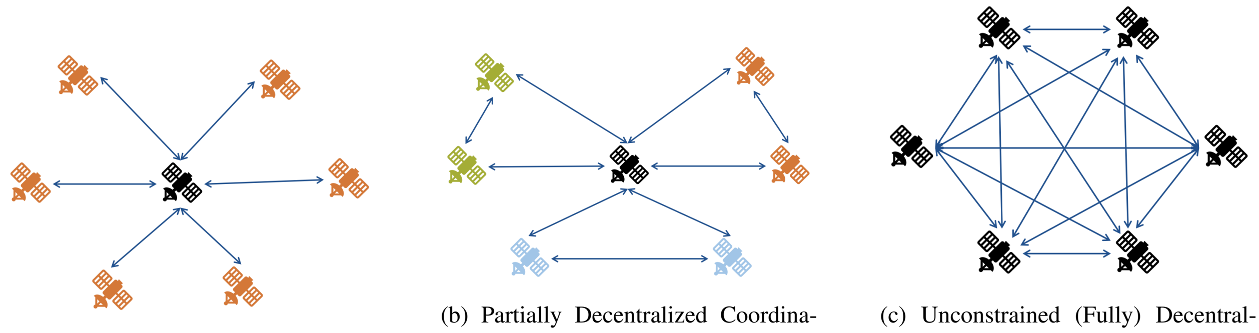
Traditional satellite constellation management relies on centralized ground-based operations using research methods that can be categorized into exact and non-exact approaches. For small-scale problems, classical exact methods like MIP and Constraint Programming provide optimal solutions but lack scalability for large constellations[10,11] . Non-exact classical methods, including heuristic approaches such as Greedy Algorithm, Genetic Algorithms, Simulated Annealing, and metaheuristics like Tabu Search and Ant Colony Optimization[12] , offer better scalability at the cost of optimality guarantees. Recent research has explored learning based techniques and RL for satellite scheduling, with studies like the one from Herrmann et al.[13] demonstrating competitive or superior performance compared to traditional methods, though most of these approaches remain centralized and require ground-based control.
The scalability limitations of centralized control have motivated the development of Multi-Agent Systems (MAS), where individual satellites act as autonomous agents capable of distributed decision-making[14] . MAS approaches can be broadly classified into centralized methods (with a central authority coordinating all agents) and decentralized methods (where agents make independent decisions based on local information). Within this framework, market-based approaches like Contract Net Protocols and Multi-Agent Reinforcement Learning (MARL) have emerged as promising solutions. Unlike single-agent RL, MARL addresses the coordination of multiple learning agents, where each satellite learns optimal policies while adapting to the concurrent learning of other agents in the system[9] . Zilberstein et al.[6] employ constraint-based heuristics for scalable decentralized scheduling, while Messina et al.[15] present consensus-based task allocation methods. However, most existing MARL applications in satellite systems face limitations in handling resource constraints and scaling beyond fixed constellation parameters.

3. Method

Our approach aims to decentralize decision-making and scheduling within the management of large constellations, leveraging the capabilities of Deep Reinforcement Learning (DRL), which is the branch of RL that builds models on deep neural networks (DNNs). This decentralized approach eliminates the need for a centralized model, allowing

DOI: 10.13009/EUCASS2025-248

individual satellites to autonomously adapt and optimize their operations in real-time. By utilizing DRL, each satellite in the network can learn and make decisions onboard based on local observations and interactions with others, leading to a more flexible and resilient system.
Our problem space consists of a LEO constellation of small satellites, acting as collaborative agents, equipped with imaging and inter-satellite link (ISL) capabilities. Their environment is populated with other orbiting objects in their vicinity. Their main objective is to locate and track these objects and spread the collected information among them to improve SSA. This aims to provide better outcome information and monitoring of space objects while optimizing the use of resources within the constellation. This optimization of shared resources and onboard capabilities is especially interesting in small satellites with limited onboard resources that perform missions with opportunistic events. Moreover, three coordination configurations are implemented in the simulation, depicted in Figure 1.



(a) Centralized Coordination

(b) Partially Decentralized Coordination

(c) Unconstrained (Fully) Decentralized Coordination

Figure 1: Comparison of Satellite Coordination Models

3.1 RL Framework

Markov Decision Processes (MDPs)[16] assume full observability, meaning agents have access to the complete state of the environment. However, in many practical scenarios, including satellite operations, agents often operate under partial observability, where they cannot access all relevant state information. This leads to the use of Partially Observable Markov Decision Processes (POMDP) problem space formulations, which are extensions of MDP designed to handle situations where the agent must make decisions based on incomplete or uncertain information, described as observations.
In a POMDP, the agent maintains a belief state, $b(s)$, which is a probability distribution over possible states based on the history of actions and observations. The objective is to find a policy $\pi : B \rightarrow A$ that maximizes the expected cumulative reward, where B is the set of all possible belief states, and A the set of actions chosen.
In the context of autonomous constellation operations, each satellite must operate with its own information about the environment. For instance, a satellite may not always have the same information or current status of another satellite due to limited communication windows or sensor constraints. This scenario fits well within the decentralized partially observable Markov Decision Process (Dec-POMDP) framework, as it inherently handles the challenges of partial observability and the need for decentralized decision-making.
Dec-POMDPs account for the fact that each agent has only partial information and must make decisions that consider the potential actions of other agents. This adds layers of complexity due to the need for coordination and the increased computational challenges, as Dec-POMDPs are known to be of high computational complexity $(2^{n^{O(1)}})$[17] .
A multi-agent Dec-POMDP is formally defined by the tuple listed below $(N, S, \{A_i\}, P, \{R_i\}, \gamma, \{O_i\}, Z)$, to which we include targets (T) for completeness. Notably, the transition probability function (P), and the observation probability function (Z) are key theoretical components. While not explicitly modeled as probability distributions, they are embedded in the simulation logic, as the probability of state and observation transitions is affected by simulation propagations. The discount factor (γ) is a predefined parameter that influences the weighting of future rewards in the cumulative reward calculation, encouraging long-term planning. RL algorithms implicitly handle these elements during training, so that the transition probabilities are effectively learned by the agent through interactions with the environment, as the agent updates its policy based on observed state transitions and rewards. Each component of this tuple plays a specific role in our satellite coordination system:

- N represents the number of agents in the system, corresponding to the observer satellites. For our case, 20 agents were used.

DOI: 10.13009/EUCASS2025-248

- T denotes the number of objects in orbit, acting as target satellites in the simulation. In our setup, 100 target satellites were simulated.
- S is the set of possible states of the environment, including all relevant variables such as satellite positions, velocities, and communication statuses.
- $\{A_i\}$ denotes the set of actions available to each agent i. There are three available actions per timestep per agent. These actions account for the activation of a flight mode, emulating a real-time scheduling process. The first option represents idle mode, the second one tries to propagate the information throughout the communication mode, and the last one accounts for activation of the payload, named observation mode.
- $\{R_i\}$ is the reward function for each agent i, providing feedback based on the actions taken and their outcomes. This work presents four different reward functions, designed to align with mission objectives by associating rewards with communicating with other observers or obtaining high-quality images of targets.
- $\{O_i\}$ represents the set of observations available to each agent i, as a subset of S, which includes partial information about the state of other satellites and targets depending on their interactions.
- P is the transition probability function, describing the probability of moving from one state to another given the current state and the actions of all agents. This encapsulates the physical dynamics of the satellites and their interactions.
- γ is the discount factor, reflecting the importance of future rewards compared to immediate ones. A higher value emphasizes long-term planning. The chosen values after hyperparameter tuning range from 0.966 to 0.98, as we try to encourage mission accomplishment in the long run.
- Z is the observation probability function, detailing the likelihood of receiving a particular observation given the state and actions. This accounts for the partial and uncertain nature of information in the system.

The objective here is to find a set of policies $\{\pi_i\}$ (Equation 1) that maximize the expected cumulative reward for all agents:

$$\{\pi_i^*\} = \arg\max_{\{\pi_i\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{i=1}^{N} R_i(s_t, a_{i,t}) \mid \{\pi_i\}\right] \tag{1}$$

The observations gathered by each agent are summarized in Table 1. The actual availability of each parameter depends on each agent's current communications and observations. Therefore, these observations are dynamic and partial, following the problem space formulation.

Table 1: Observation space parameters available to each Observer Satellite

Parameter	**Description**	**Data Type**
observer_satellites	Own and contacted observer satellites orbital parameters	Continuous
band	Own communication band identifier (1–5)	Discrete
availability	Own availability status	Binary
target_satellites	Tracked target satellites' position and velocity	Continuous
battery	Battery level of observers	Continuous
storage	Storage level of observers	Continuous
observation_status	Observation status of target satellites	Discrete
communication_status	Communication status between observers	Binary

Making use of this framework, we leverage centrally trained, decentralized executed agents for real-time decision-making onboard the satellites studying the effect of four different reward functions, as presented in Section 3.3, with actions taken only based on limited, rapidly changing information.

3.2 Simulation

This part is fundamental for accurately simulating the physical and functional attributes of the environment. As mentioned in previous sections, there are two primary types of satellites participating in our custom simulation: observer

DOI: 10.13009/EUCASS2025-248

satellites, responsible for gathering data, observations, and making decisions based on them, and target satellites, which serve as orbiting space objects.
The observation status of the targets comprises undetected (0), detected (1), being observed (2), and observed (3). The final mission accomplishment is based on the average statuses that each agent has on the targets. In short, a perfect mission score is a mission in which all observer satellites have all targets marked as observed. This can be achieved either by direct observation or by obtaining the information via communication from another observer of the constellation. In contrast, a simulation with all targets marked as undetected has a 0% mission score.
Observer satellites are deployed using a Walker-Delta constellation with 5 orbital planes, with an orbit height between 700 km and 800 km, and 56° of inclination. They are initialized in the simulations with arbitrary battery and storage levels from 20% to 70%. We assume a small battery, with 80 Wh, and different power consumption rates for each operational mode. For the storage system, we assume a total storage of 32 GB, and the size of captured data and messages sent varies depending on the time per observation or the amount of novel information they share. In Table 2, a short summary of the satellite configuration can be found.

Table 2: Observer Satellites Parameters

Parameter	**Value/Range**	**Units**
Orbital Parameters		
Semi-major axis	7 070 – 7 170	km
Inclination	56.0	degrees
Eccentricity	0.001	–
Orbital planes	5	–
RAAN	Distributed by plane	degrees
Argument of perigee	0.0	degrees
Power System		
Maximum energy storage	80	Wh
Solar panel area	0.1	m^2
Solar panel efficiency	0.3	–
Power consumption (standby)	7.5	W
Power consumption (communication)	9	W
Power consumption (observation)	19	W
Data & Communication		
Maximum data storage capacity	32	GB
Observation data rate	1	Mbits/s
Communication bands	1 – 5[a]	–
Optical Payload		
Aperture diameter	0.09	m
Field of view	10	degrees

[a]Depending on coordination type

Regarding targets, they have orbit heights ranging from 600 km to 900 km, with randomized orbital parameters around Sun-Synchronous Orbit (SSO), where the highest density of satellites and debris is for these orbits[18] , but without following any patterns or specific formation. They have an equivalent size of 1 m. A summary of the initial target parameters can be found in Table 3.

Table 3: Target Satellites Parameters

Parameter	**Value/Range**	**Units**
Orbital Parameters		
Semi-major axis	6 970 – 7 270	km
Inclination	SSO [a]	degrees
Eccentricity	0 – 0.001	–
RAAN	0 – 180	degrees
True anomaly	0 – 360	degrees

[a]With random variation between ±5°

To ensure realistic and reliable simulation outcomes, the orbit propagation has been validated against the Astropy

DOI: 10.13009/EUCASS2025-248

library[19] . Communication and observation procedures are checked using precomputed positions and attitudes to verify that these actions only take place in the correct situations, emulating situations in the border limits of ranges in which they can take place.
Observer satellites are equipped with an optical payload featuring a 0.09 m aperture diameter, suitable for small satellite platforms including CubeSats, and a 10° field of view, as depicted in Figure 2. This configuration enables detection of 1-meter objects up to a maximum range determined by the diffraction limit. The maximum observable distance for a given object size, derived from the Rayleigh criterion, is:

$$D_{\max} = \frac{a \cdot d}{2.44 \cdot \lambda} \tag{2}$$

where $D_{\max}$ is the maximum observable distance in meters, a is the object size in meters, d is the aperture diameter in meters, and λ is the wavelength of light in meters. Using a wavelength of 700 nm (upper limit of visible light), the maximum observable distance of 52.7 km is obtained for 1-meter targets.

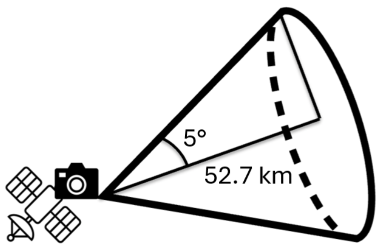


Figure 2: Field of view of observer satellites.

During observing mode activation, the satellite's power consumption increases to 19 W. The system scans all targets within the 10° field of view, applying the 52.7 km detection range constraint. Among detectable targets, the system selects the one with optimal pointing accuracy to maximize observation quality. The resulting data is stored locally and flagged for constellation-wide sharing during future communication opportunities. The observation logic is summarized in Algorithm 1.

Algorithm 1: Observation Logic

```
1: Initialize steps ← 0
2: Get power_consumption ← power_consumption_rates["observation"]
3: Initialize target ← -1, best_pointing_accuracy ← 0
4: for each target in targets do
5:     if already observed target then
6:         continue {Skip already observed targets}
7:     end if
8:     Calculate distance ← distance_between(target, time_step)
9:     Calculate pointing_accuracy ← evaluate_pointing_accuracy(target, time_step)
10:    if distance ≤ maximum distance and pointing_accuracy > best_pointing_accuracy then
11:        best_pointing_accuracy ← pointing_accuracy
12:        best_target ← target
13:    end if
14: end for
15: if best_target ≥ 0 then
16:     Observe the best target
17:     Update internal matrices
18:     Update Battery level
19:     Calculate reward("observation", observer, observation_result)
20:     reward ← reward("observation", observer, observation_result)
21: else
22:     No observable targets
23:     reward ← 0
24: end if
```

DOI: 10.13009/EUCASS2025-248

Communications on the observers emulate UHF omnidirectional antennas, with a maximum range based on the capabilities of the components stated in Table 2, without attitude dynamics nor Doppler effect. An effective data rate is computed based on antenna and transceiver powers, gains, and losses, assuming QPSK modulation, Bit Error Rate (BER), and Free Space Loss (FSL) effects[20] . For a communication to be successful, the transmitter has to activate the communication mode, and the receiver has to be within range and in idle mode. If a receiver observer is processing an observation or not available, the communication fails.
Communication bands can also be assigned to each observer, allowing a way of creating different network topologies in the simulation. The are three coordination topologies implemented in the simulator: centralized, fully decentralized, and constrained decentralized. In the centralized version, one random observer is initialized as a central node, being the only one in the constellation able to propagate the information it receives from the rest of the constellation. For the fully decentralized coordination, every node can communicate with each other, while for the constrained decentralized configuration, satellites can only communicate between them if they share the same communication band 4 alternatives. Moreover, communication can take place during more than one single timestep, accounting for large-sized packages being sent. If the communication is interrupted or the signal is lost, the communication is assumed to have failed. A pseudo-code formulation of the communication is presented in Algorithm 2.

Algorithm 2: Communication Process

```
1: Initialize communication_done ← False, max_steps ← 0
2: Get power_consumption ← power_consumption_rates["communication"]
3: Calculate data_to_transmit ← current orbital parameters + contacted observers + observed targets
4: for each other_observer in observer_satellites do
5:    if other_observer == observer then
6:       continue {Skip self communication}
7:    end if
8:    communication_done ← False
9:    steps ← 0
10:   data_transmitted_this_timestep ← 0
11:   total_data_transmitted ← 0
12:   while not communication_done and total_data_transmitted < data_to_transmit do
13:      Try propagate_information(i, other_observer, j, . . . )
14:      total_data_transmitted += data_transmitted
15:      if communication_done or data_transmitted_this_timestep == 0 then
16:         Update internal matrices
17:         break
18:      end if
19:   end while
20:   max_steps ← max(steps, max_steps)
21: end for
22: Calculate reward("communication", observer, communication_result)
23: reward ← reward_function.calculate_reward("communication", observer, communication_result)
24: Update internal matrices
25: Update Battery level
```

3.3 Reward Functions

The success of the RL techniques heavily depends on the effective reward function (RF) design. This work implements a Proximal Policy Optimization (PPO) algorithm with four different reward functions to determine which type leads to better learning policies. They are designed taking into account the resources and goals of the constellation, each of them having three main components: resources, actions, and mission goal. Additionally, a sensitivity analysis of scaling factors of the mission goal term is carried out to see the effect on final mission results.
Presented in Table 4 is a list of parameters defined to shape the reward functions. The scaling factor values are adjusted as suggested in literature[9,21] , so that the total return per agent has an order of magnitude not too small for gradients to be significant enough, nor too big to avoid unstable behaviours.
In all presented cases, a common evaluation of the resources and the direct consequences of actions is made using the terms in Equations 3 and 4:

DOI: 10.13009/EUCASS2025-248

Table 4: Observation and Reward Function Parameters

Parameter	**Description**
o_t^i	Observation at time t for observer i
a_t^i	Action at time t for observer i
$M_{i,j}$	Observation status of target j by observer i
T	Total number of targets
σ	General reward scaling factor
k	Resource consumption penalty factor
p_{standby}	Standby penalty
ρ	Communication success reward factor
$R_{\max}$	Communication reward cap
μ	Failed action penalty factor
r_{obs}	Fixed observation reward
λ	Quality of observation reward factor
E	Energy level (from observations o_t^i)
S	Storage level (from observations o_t^i)
$d_{\text{transmitted}}$	Data transmitted (from observations o_t^i)
	Sensitivity Analysis
α	Scaling Factor for mission state bonus

Resource Penalty:

$$R_{\text{resource}}(o_t^i) = -k\left(\frac{E_{\max} - E_{\text{available}}}{E_{\max}} + \frac{S_{\max} - S_{\text{available}}}{S_{\max}}\right) \tag{3}$$

Action-specific Rewards:

$$R_{\text{action}}(a_t^i) = \begin{cases} -p_{\text{standby}} & \text{if } a_t^i = \text{standby} \\ \min(\rho \cdot d_{\text{transmitted}}, R_{\max}) & \text{if } a_t^i = \text{communication, successful} \\ -\mu & \text{if } a_t^i = \text{communication, failed} \\ r_{\text{obs}} + \lambda \cdot \text{Quality} & \text{if } a_t^i = \text{observation, successful} \\ -\mu & \text{if } a_t^i = \text{observation, failed} \end{cases} \tag{4}$$

To account for resource utilization and to encourage efficient energy and storage management, a proportional penalty is given based on available resources. The communications reward is proportional to the amount of data sent, up to a maximum reward cap. The observations are evaluated based on the quality of the observation, plus a small fixed reward for each one.
In addition to the resource and action-specific rewards, the four reward functions implement an additional term to evaluate the mission goal, as can be seen in Equation 5. The first and second reward function variations can be seen as individual evaluations for the cooperative constellation mission, taking only individual observations into account (Equations 6, 7), while the third and fourth reward functions evaluate the performance based on the collective achievements of the whole constellation (Equations 8, 9). Moreover, each pair of reward functions (1 - 2 and 3 - 4) has positive and negative variations of both the individual and collective reward function approaches, resulting in a total of four variations.

$$R^i_{total}(o_t^i, a_t^i, M) = R_{\text{resource}}(o_t^i) + R_{\text{action}}(a_t^i) \pm R_{mission}(M) \tag{5}$$

3.3.1 Reward Function 1: Individual Positive Reinforcement

Positive rewards for monitored targets based on individual observation matrices:

$$R_1^i(o_t^i, a_t^i, M_{i,j}) = \sigma\left[R_{\text{resource}}(o_t^i) + R_{\text{action}}(a_t^i) + \alpha \cdot \frac{|\{j : M_{i,j} > 0\}|}{T}\right] \tag{6}$$

3.3.2 Reward Function 2: Individual Negative Reinforcement

Negative rewards for undetected targets based on individual observation matrices:

$$R_2^i(o_t^i, a_t^i, M_{i,j}) = \sigma\left[R_{\text{resource}}(o_t^i) + R_{\text{action}}(a_t^i) - \alpha \cdot \frac{|\{j : M_{i,j} = 0\}|}{T}\right] \tag{7}$$

DOI: 10.13009/EUCASS2025-248

3.3.3 Reward Function 3: Collective Positive Reinforcement

Individual terms for resources and actions, plus a positive global collective mission goal term based on all observation status matrices:

$$R_3^i(o_t^i, a_t^i, M_{i,j}) = \sigma\left[R_{\text{resource}}(o_t^i) + R_{\text{action}}(a_t^i) + \alpha \cdot |\{j : \exists i \text{ s.t. } M_{i,j} = 3\}|\right] \tag{8}$$

3.3.4 Reward Function 4: Collective Negative Reinforcement

Individual terms for resources and actions, plus a positive global collective mission goal term based on all observation status matrices:

$$R_4^i(o_t^i, a_t^i, M_{i,j}) = \sigma\left[R_{\text{resource}}(o_t^i) + R_{\text{action}}(a_t^i) - \alpha \cdot |\{j : \forall i, M_{i,j} \neq 3\}|\right] \tag{9}$$

To assess the effect of the bonus parameters on the final mission goal, α is varied to perform a sensitivity analysis on the scaling factor of the mission bonus term. α can be seen as the weight of this mission term compared to the resources and actions' rewards. Six α values are explored: $[0, 0.1, 0.5, 1, 2, 5]$, with a value of 0 assigning no reward for mission progression, 1 corresponding to the mission term having the same importance as resources and immediate actions terms, and bigger values of 2 and 5 giving more relative weight to the final mission goal.

4. Results

To evaluate the performance of the proposed RL-based approach, two deterministic benchmark policies are implemented as comparison baselines. These benchmark policies operate as centralized methods with perfect information access, having complete knowledge of all system states, including battery levels, storage capacities, target positions, and communication statuses across the entire constellation. This contrasts with the RL agents that operate under partial observability and decentralized decision-making. The rule-based policy implements a hierarchical decision-making strategy using predefined thresholds for battery and storage levels. This policy prioritizes energy conservation when battery levels fall below 30%, encourages communication when storage exceeds 70%, and promotes observation when battery levels are high and storage is low. The Mixed Integer Programming (MIP) policy formulates the action selection as an optimization problem that maximizes expected utility while respecting resource constraints. Due to computational complexity, this implementation uses a simplified approach with linear programming relaxation as a heuristic, calculating utilities for idle, communication, and observation actions based on the current system state and selecting the action with the highest utility. Both benchmark policies operate using the same three-action space as the RL agents (idle, communicate, observe) and are evaluated across all coordination configurations to provide comprehensive performance comparisons against the learned policies.
RL trainings were carried out using 5 randomization seeds for 100 iterations for each of the 4 cases, with 6 different α scaling factors ranging from 0 to 5, in the fully decentralized coordination environment. Each one of the 120 experiment variations used over 30 M agent steps (more than 1.6 M environment steps) to train the policies. In other related works, 50 M training steps were used in their PPO policy[5] . Due to the number of experiments carried out, the time taken per iteration in our simulator, and the computing time limitations of the service provider, the limitation of 100 iterations and 5 seeds per run was chosen. Figure 3 depicts the results of the training, displaying the average value and deviations of the 5 different training seeds for each case. Here, the complexity of reward function design can be seen by noticing that directly increasing the scaling factor does not necessarily lead to higher returns, with α values of 0.1 and 1.0 reaching the highest rewards during training instead of the 2.0 and 5.0 cases.
Results summarized in Figure 4 show the evaluation of 15 simulations per coordination topology, scaling factor, and reward function type, using different seeds from the training ones. For all experiments, the seed with the greatest returns in training is the one chosen. Mission accomplishment is calculated based on the average observation status of targets among all agents, thus accounting for both the observation and the communication performance. The results demonstrate a hierarchy across coordination scenarios: centralized coordination achieves mission accomplishment rates higher than 60% for most cases, reaching values of up to 100%; constrained decentralized coordination shows performances between 35-75%, while fully decentralized coordination drops to 30-45% for most cases.
A striking observation from the trained policies is their superior performance when deployed in coordination scenarios different from the fully decentralized environment in which they were trained. For instance, policies trained in fully decentralized coordination achieve mission accomplishment rates 1.2-2x higher when deployed in centralized coordination, with some configurations reaching performance levels comparable to or exceeding the benchmark exact methods. This transfer learning capability suggests that the RL agents develop fundamental coordination strategies that are robust across different network topologies. The Individual Positive reward function demonstrates particular effectiveness in this cross-deployment scenario, maintaining high performance across all coordination levels for α=1.0. From their side, the benchmark methods, while maintaining consistent performance across coordination scenarios, are

DOI: 10.13009/EUCASS2025-248

Table 5: PPO Algorithm Configuration Summary

Parameter	**Value**
Learning Rate (lr)	0.0001-0.0002
Discount Factor (γ)	0.966-0.98
Train Batch Size	4096
Minibatch Size	256
Number of Epochs	30
Rollout Fragment Length	128
Framework	PyTorch
Hidden Layers	[256, 256]
Activation Function	tanh
Number of Targets	100
Number of Observers	20
Duration	86400[a]
Reward Type	1-4

[a]Duration in seconds (24 hours)

constrained by their reliance on perfect information access and centralized execution, making them less realistic for deployment in real large-scale distributed constellations.

The sensitivity to the α scaling factor reveals characteristics of the different reward formulations. Individual-based reward functions show contrasting behaviors: Individual Positive (IP) exhibits high sensitivity to α variations with performance differences of up to 60% between α=0.0 and α=1.0 in centralized scenarios, while Individual Negative (IN) demonstrates more stability across α values. Similarly, collective reward functions show this pattern, with Collective Negative (CN) maintaining more consistent performance than Collective Positive (CP) across different α settings. This stability of negative reward functions suggests they inherently provide more robust policies that are less sensitive to the mission-resource balance parameter, making them potentially more suitable for mission scenarios where the optimal balance between exploration and resource conservation is unknown a priori.

Resource utilization analysis shows that RL approaches achieve comparable or slightly superior resource conservation compared to benchmark methods, with remaining resources consistently maintained at 60-70% across all coordination topologies and reward functions. The stability of resource levels across different α scaling factors indicates that the learned policies effectively balance mission execution with resource preservation.

The difference in operating mode selection between fully decentralized, constrained decentralized, and centralized cases reveals distinct behavioral patterns across coordination topologies. In centralized coordination, RL methods demonstrate diverse action distributions with balanced use of observe and communicate actions, particularly evident in Individual Positive (IP) and Individual Negative (IN) cases where observation payload activities comprise 30-50% of total mission time. However, as coordination becomes more decentralized, all methods converge toward higher idle percentages, with fully decentralized scenarios showing 85-95% idle time across most configurations. The benchmark methods (Rule-Based and MIP) maintain consistently high idle percentages in the range of 95-98% across all coordination levels, suggesting conservative resource management strategies. Notably, negative reward functions (IN, CN) maintain more balanced action distributions even in constrained and centralized scenarios, with observation percentages remaining around 20% of mission time in constrained decentralized coordination, demonstrating their ability to maintain mission-critical activities under communication limitations.

The more stable behaviour of negative reward functions aligns with the work presented by Müller[21] , which suggests shifting the potential functions of rewards towards values equal to or lower than zero leads to faster convergence and better optimal decisions. Our reward functions were deliberately designed to produce negative values in most of the timesteps to enhance exploration, and the results demonstrate that this approach not only drives thorough exploration during training but also produces more robust, stable and generalizable policies.

To validate the feasibility of onboard deployment, the trained policies were evaluated on an NVIDIA Jetson Orin Nano operating in 7 W power mode, a configuration specifically designed for resource-constrained applications. This computing platform has been identified as a promising candidate for AI-enabled nanosatellite missions due to its balance of size, computational capability and power efficiency[22] . Performance evaluation consisted of continuous inference testing over extended periods, measuring the time required for action selection given observation inputs. The empirical results demonstrate an average inference time of approximately 50 μs per decision, confirming that the developed neural network policies can be executed in real-time on low-power onboard systems without compromising mission-critical response times. This computational efficiency, combined with the 7W power consumption, makes the

DOI: 10.13009/EUCASS2025-248

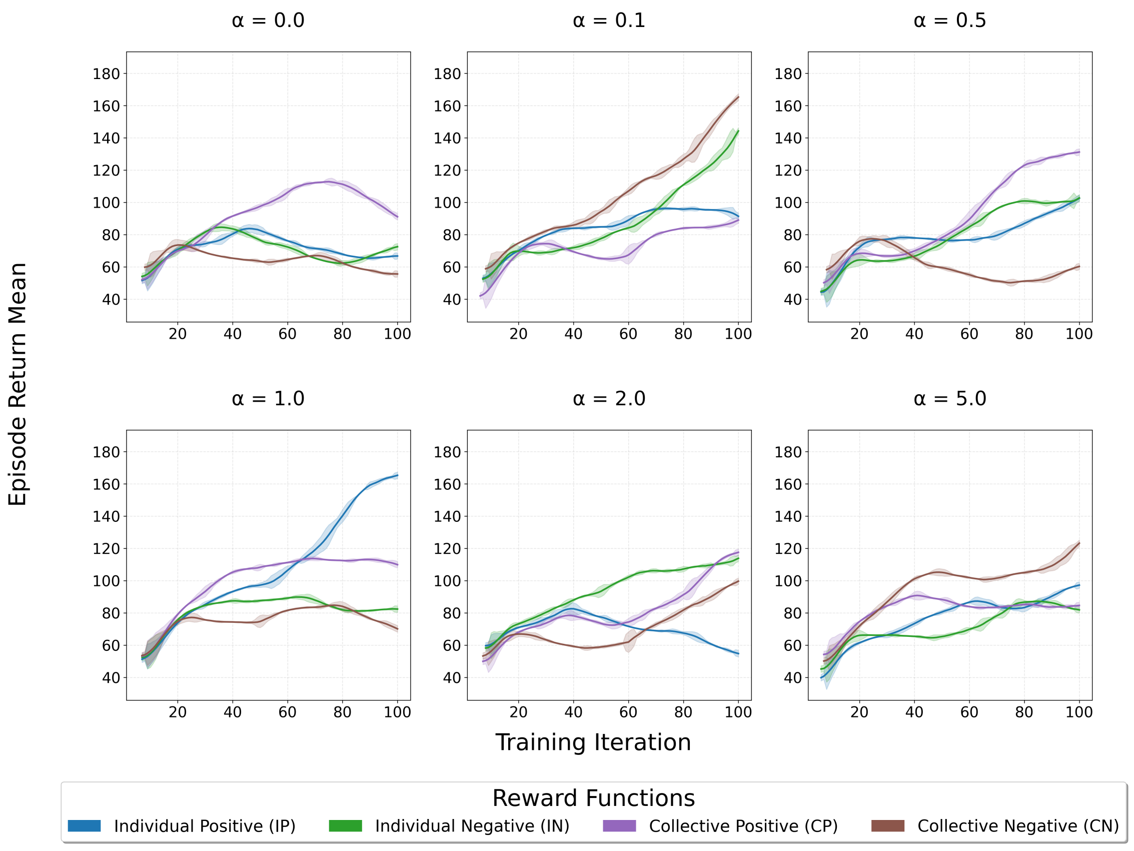


Figure 3: Training performance across different scaling factor values.

approach viable for deployment in power-constrained satellite platforms.

5. Conclusion

This research contributes to the growing field of autonomous space systems by demonstrating that decentralized, on-board AI-driven approaches can reduce dependence on ground-based control systems for large SSA satellite constellations. The findings show that multi-agent reinforcement learning provides a viable framework for spacecraft coordination problems, with negative reinforcement strategies offering superior stability and robustness in resource-constrained, mission-critical environments. The demonstrated transfer learning capabilities and reward function stability represent important steps toward more autonomous satellite constellation operations, offering a pathway to more scalable, efficient, and robust space systems that can adapt to varying and opportunistic operational conditions.
This work presents an application of decentralized multi-agent deep reinforcement learning for autonomous satellite constellation management in collaborative space situational awareness missions under dynamic and uncertain conditions. We formulated the satellite constellation coordination problem using a Dec-POMDP framework, enabling realistic modeling of partial observability and communication constraints inherent in space environments. The comprehensive evaluation of four distinct reward functions and a sensitivity analysis of the mission goal reward term reveals insights about reward function design and coordination strategies for satellite constellations.
The approach addresses scalability limitations of traditional centralized satellite operations by demonstrating effective coordination for a constellation of 20 observer satellites tracking 100 target objects. RL methods achieve comparable or superior resource conservation compared to benchmark approaches, maintaining 60-70% remaining resources across all coordination topologies while adapting their operational behavior to communication constraints. Action distribution analysis reveals that negative reward functions maintain more balanced operational modes even under communication limitations, with observation activities remaining above 20% in constrained scenarios.
Hardware validation on an NVIDIA Jetson Orin Nano operating using 7 W power mode confirms practical feasibility for current and future satellite missions, with inference times in the order of microseconds per decision, meeting the computational constraints of modern space systems. This computational efficiency, combined with low power consumption, makes the approach viable for real-time deployment in power-constrained satellite platforms.

DOI: 10.13009/EUCASS2025-248

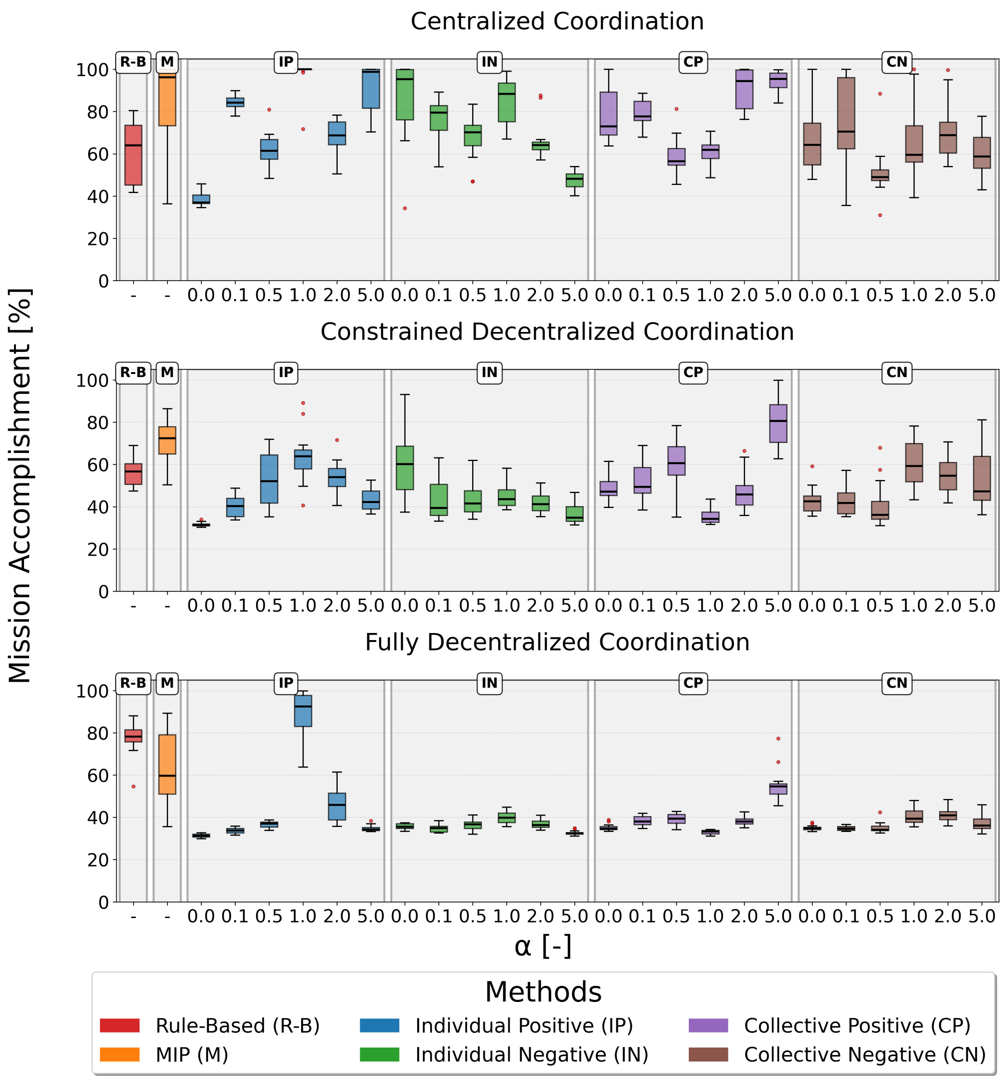


Figure 4: Mission accomplishment in percentage for 15 simulations, N=20, T=100.

6. Acknowledgments

The authors gratefully acknowledge the computational and data resources provided by the Leibniz Supercomputing Centre (łwww.lrz.de). This work was supported by the AUTOPS project, within the Bavarian Joint Research Program of the StMWi (BayVFP), funding number MRF-2307-0004.

References

[1] ESA Space Debris Office. ESA's annual space environment report. Technical Report GEN-DB-LOG-00288-OPS-SD, European Space Agency, Darmstadt, Germany, July 2024. Issue/Revision 8.0.

[2] Yoke T Yoon, Paolo Ghezzo, Calum Hervieu, and Ignacio Dominguez-Adame Palomo. Navigating a large satellite constellation in the new space era: An operational perspective. *Journal of Space Safety Engineering*, 10(4):531–537, 2023.

[3] Nitinder Mohan, Andrew E. Ferguson, Hendrik Cech, Rohan Bose, Prakita Rayyan Renatin, Mahesh K. Marina, and Jörg Ott. A multifaceted look at starlink performance. In *Proceedings of the ACM Web Conference 2024*, pages 2723–2734. ACM, May 2024.

[4] Gauthier Picard, Clément Caron, Jean-Loup Farges, Jonathan Guerra, Cédric Pralet, and Stéphanie Roussel. Autonomous agents and multiagent systems challenges in earth observation satellite constellations. In *Proceedings*

DOI: 10.13009/EUCASS2025-248

of the 20th International Conference on Autonomous Agents and MultiAgent Systems, pages 39–44, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[5] Mark A. Stephenson and Hanspeter Schaub. Bsk-rl: Modular, high-fidelity reinforcement learning environments for spacecraft tasking. In *75th International Astronautical Congress (IAC)*, Milan, Italy, October 2024.

[6] Itai Zilberstein, Ananya Rao, Matthew Salis, and Steve Chien. Decentralized, decomposition-based observation scheduling for a large-scale satellite constellation. *Journal of Artificial Intelligence Research*, 82:169–208, January 2025.

[7] James Boerkoel, James Mason, Daniel Wang, Steve Chien, and Adrien Maillard. An efficient approach for scheduling imaging tasks across a fleet of satellites. Technical Report 2014/55086, Jet Propulsion Laboratory, National Aeronautics and Space Administration, Pasadena, CA, 2021.

[8] Sreeja Nag, Alan S. Li, and James H. Merrick. Scheduling algorithms for rapid imaging using agile cubesat constellations. *Advances in Space Research*, 61(3):891–913, 2018.

[9] Stefano V Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press, 2024.

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

[11] Xiaoyu Chen, Gerhard Reinelt, Guangming Dai, and Andreas Spitz. A mixed integer linear programming model for multi-satellite scheduling. *European Journal of Operational Research*, 275(2):694–707, June 2019.

[12] Benedetta Ferrari, Jean-François Cordeau, Maxence Delorme, Manuel Iori, and Roberto Orosei. Satellite scheduling problems: A survey of applications in earth and outer space observation. *Computers & Operations Research*, page 106875, 2024.

[13] Adam Herrmann and Hanspeter Schaub. A comparative analysis of reinforcement learning algorithms for earth-observing satellite scheduling. *Frontiers in Space Technologies*, 4, November 2023.

[14] Jonathan Bonnet, Marie-Pierre Gleizes, Elsy Kaddoum, Serge Rainjonneau, and Gregory Flandin. Multi-satellite mission planning using a self-adaptive multi-agent system. In *2015 IEEE 9th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 11–20. IEEE, September 2015.

[15] Vincenzo Messina and Alessandro Golkar. Advancing federated satellite systems performance: A collaborative method for improved object detection in space. In *AIAA SCITECH 2025 Forum*, Reston, Virginia, January 2025. American Institute of Aeronautics and Astronautics.

[16] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley Sons, Inc., USA, 1st edition, 1994.

[17] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, May 2016.

[18] Romain Lucken. Systematic space debris collection using Cubesat constellation. *Proceedings of the 7th European Conference for Aeronautics and Space Sciences*, page 13 pages, 2017.

[19] Astropy Collaboration. The astropy project: Sustaining and growing a community-oriented open-source project and the latest major release (v5.0) of the core package. *The Astrophysical Journal*, 935(2):167, 2022.

[20] J. R. Wertz and W. Larson. *Space Mission Analysis and Design*. Springer Netherlands, 1999.

[21] Henrik Müller and Daniel Kudenko. Improving the effectiveness of potential-based reward shaping in reinforcement learning. *arXiv preprint arXiv:2502.01307*, 2025.

[22] Lara Schuberth, Vincenzo Messina, Ramón María García Alarcia, Jaspar Sindermann, Kian Bostani Nezhad, Roberto Aldea Velayos, Sofia Lennerstrand, Julie Rollet, Federico Sofio, Alessandro Tinucci Monibas, et al. Leveraging event-based cameras for enhanced space situational awareness: A nanosatellite mission architecture study. In *75th International Astronautical Congress (IAC)*, Milan, Italy, October 2024.