AIAA SciTech Forum
12-16 January 2026, Orlando, FL
AIAA SCITECH 2026 Forum

10.2514/6.2026-0131

Multi-Agent Reinforcement Learning for Swarm Planetary Exploration

A. Menor de Oñate* and E. van Kampen†
Delft University of Technology, P.O. Box 5058, 2600GB Delft, The Netherlands

Exploring planetary bodies using robot swarms can potentially increase the value of the exploration missions; enabling the execution of novel measurements and explorations previously deemed impractical or unattainable. Despite its potential, the technology readiness level of planetary swarms is not very mature. This work uses multi-agent reinforcement learning to find control policies that allow swarms to autonomously explore unknown areas in a decentralized manner, contributing towards the technology readiness of the field. A multi-agent proximal policy optimization (MAPPO) algorithm is proposed for this end, where the policy uses LIDAR perception information, and the input of the value function contains local and global environment information. The algorithm finds control policies that achieve cooperation behaviors and generalize to unseen swarm sizes and environments learning with simple, sparse reward functions. Moreover, different types of reward functions, value inputs, and environment configurations are investigated.

Nomenclature

a.u.	=	Arbitrary Units
CNN	=	Convolutional Neural Network
CTDE	=	Centralised Training, Decentralised Execution
DEC-MDP	=	Decentralised Markov Decision Process
DEC-POMDP	=	Decentralised Partially Observable Markov Decision Processes
IPPO	=	Individual PPO
LIDAR	=	Light Detection and Ranging
MaCMAS	=	Methodology Fragment for Analysing Complex Multi-agent Systems
MAPPO	=	Multi Agent PPO
MARL	=	Multi-Agent Reinforcement Learning
MDP	=	Markov Decision Process
ML	=	Machine Learning
POI	=	Point of Interest
POMDP	=	Partially Observable Markov Decision Process
PPO	=	Proximal Policy Optimisation
RL	=	Reinforcement Learning
TRL	=	Technology Readiness Level

I. Introduction

Over the past half century, planetary exploration missions have relied on single space vehicles [1]. These vehicles are highly complex and expensive and can compromise the exploration mission if a failure occurs.
Because of their resiliency, low cost, and adaptability, teams and swarms of autonomous robots are being considered to undertake future space exploration missions [2]. In this context, a swarm of space vehicles is a *"collection of often smaller and simpler, autonomous vehicles that coordinate in a decentralized manner to achieve a common goal"* [2]. Exploration tasks performed by a single, complex vehicle, could be done by a collection of simpler vehicles instead.

*MSc. Student, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.
†Associate Professor, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.

Swarms unveil the possibility of increasing the value of the planetary exploration missions, and *"open a new world of science exploration"* [3], allowing for performing measurements and explorations that were previously impractical or impossible; executing large spatial aperture missions such as seismological investigations, fast terrain measurements, interferometry, exploration of Moon and Mars lava tubes, etc, as recognized by the German Aerospace Center [4] and NASA's JPL [2].
As outlined in *Intelligence in Future NASA Swarm-based Missions* [5], to achieve the aforementioned missions, swarms must be **autonomous**, **intelligent**, and have the capability of **learning from their environment**.

In this regard, the technology for deploying swarm missions for planetary exploration is not fully mature. In *Space Vehicle Swarm Exploration Missions: A Study of Key Enabling Technologies and Gaps* [2] it is highlighted that most of the technology readiness levels (TRLs) for performing planetary exploration, mapping and sampling, and cooperative task recognition and allocation are not very mature (3-5 TRL), or not available/ in conceptual stages of development (1-2 TRL).

In this work, multi-agent reinforcement learning (MARL) is studied to contribute to the readiness of swarm missions. MARL belongs to the field of machine learning (ML) methods, specializing in finding control strategies that can make a collection of individual entities achieve a global goal. As such, it has the potential to learn or discover control policies that are difficult or virtually impossible to find by pre-established heuristics or priors, by interacting with a simulated environment and receiving a performance-based feedback signal (the reward function) which can be designed from the mission requirements.

However, the MARL optimization problem remains an extremely challenging task [6–8], with actor-critic MARL architectures being used in the state-of-the-art of the field. A multi-agent extension of the Deep Deterministic Policy Gradient RL algorithm has been proposed in state-of-the-art studies in MARL technologies for planetary exploration [9]. However, in this architecture, the agent training scalability is severally limited (5 agents or less), the robustness and scalability of the learned policies are not addressed, and the reward function is dense; thus heavily relying on heuristics which may limit the potential of the swarm.

This work proposes a MARL* algorithm that achieves prominent performance in the field of MARL applied to swarm planetary exploration in the domains of agent scalability, agent perception, reward function flexibility, and policy analysis while considering NASA mission requirements. The algorithm is a multi-agent extension of Proximal Policy Optimization (PPO) which uses a simulated LIDAR (Light Detection and Ranging) for the individual perception of each agent and a global environment tensor processed by a convolutional neural network modeling the critic of the agents.

The contributions of this study are summarized as follows:

- A framework is proposed to improve the intelligence of swarm missions, integrating the field of MARL with NASA swarm architectures, and targeting areas with low TRLs. Specifically, the objective is obtaining local cooperative guidance policies that achieve a global mission goal, and investigating the usage of MARL for policy discovery in complex settings. To the best of the authors' knowledge, this is the first time NASA swarm mission requirements and MARL solutions for this end have been analyzed.
- A Multi-Agent PPO algorithm specifically tailored for swarm planetary exploration is developed. Unlike other works in MARL, there is no explicit use of dense reward functions, thus potentially allowing for the implementation of reward functions derived directly from mission requirements. Furthermore, the algorithm operates in continuous action spaces and allows for homogeneous and heterogeneous agent populations (however, homogeneous swarms are considered for this work).
- A planetary swarm exploration environment is developed, which has more complexity than the environment used in [9] regarding the amount and placement of targets, agents, and obstacles, where the proposed algorithm learns local policies that can adapt to environments and swarm sizes that have not been seen during training, and show emergent behaviors.
- Different algorithm, environment, and reward function configurations are examined to assess how they affect learning performance, policy generalizability, and gain insights into the MARL problem; thus further improving the TRL of swarm exploration missions.

*https://github.com/adrianmenor/MAPPO_for_swarm_planetary_exploration

II. Background

This section provides relevant background information on MARL and, relevant prior studies are cited to offer contextual insights drawn from the aerospace and machine learning fields and highlight the focus of this research.

A. MARL

MARL applies reinforcement learning to systems with several agents. Multi-agent systems consist of a group of autonomous, interactive entities that all share a common environment, which they perceive with sensors and in which they can act through actuators. In this work, collaborative swarms are considered.

MARL scenarios have certain distinctive characteristics compared with the traditional RL paradigm. The fundamental design of MARL algorithms considers aspects from Temporal Difference methods, Game Theory, and Direct Policy Search [6, p. 13]. Classic RL theory is grounded in the assumption of MDPs. In MARL this assumption is often not valid [6], as the problem becomes a partially observable MDP (POMDP), a decentralized MDP (DEC-MDP), or as is the case in this study, a decentralized, partially observable MDP (DEC-POMDP), which is notoriously difficult to solve and can require super-exponential time in the worst case [7, 8]. This is because DEC-POMDPs model problems in which multiple agents, each possessing partial observations of the environment, make decisions simultaneously to achieve a common objective.

MARL presents some additional challenges compared to classic single-agent reinforcement learning, which results in the need to modify the structure of the learning algorithms, and the perception of the agents. Here we discuss some of the most salient characteristics/challenges of MARL:

- In multi-agent scenarios, the environment gets affected by the actions of all agents, making it non-stationary. Consequently, each agent encounters a dynamic learning challenge where the optimal policy continually shifts in response to changes in the policies of other agents.
- MARL algorithms with centralized training often suffer from the *"curse of dimensionality"* [6], where the input size of the value functions grows exponentially as the number of agents is increased since each new agent adds its own variables to the environment.
- The training stability of the agents (the convergence to a stationary policy) needs a trade-off with the adaptation capabilities with respect to the changing behavior of the other agents [6]. Thus if the policy rapidly updates to react to the other agents, it is likely to have an oscillatory performance during training.
- In accordance with the aforementioned information, the overall learning stability of MARL algorithms remains a challenging task [6].
- MARL settings are susceptible to fall in local minima (which can be related to fall in a suboptimal Nash equilibrium).
- In this study sparse reward functions are allowed (since we propose a framework to find control policies in situations where the problem information is limited), and this can further hinder learning performance.

MARL algorithms typically lie within two frameworks: centralized and decentralized learning [10]. Here both options are explored; the proposed algorithm has a policy that only perceives a local state of the environment, and value functions that can contain the global information during training. Combining a local policy with a global value function is referred to as **Centralized Training, Decentralized Execution** (CTDE). CTDE is usually selected as the approach to solving the DEC-POMDP in state-of-the-art MARL algorithms [11], especially in actor-critic architectures.

Having value functions that use some form of global information helps relax the issue of the environment being non-stationary, but if this global information scales poorly as more agents are added to the environment, the input to the value function becomes too large, leading to the aforementioned curse of dimensionality.

B. Related Work

In recent research, the field of MARL has witnessed significant progress. Specific to planetary and space exploration, [9] showcased the effectiveness of a multi-agent deep deterministic policy gradient (MADDPG) algorithm, achieving successful learning outcomes in simple planetary exploratory tasks with five agents or less, and dense reward functions. Additionally, NASA explored deep Q-learning and Advantage Actor-Critic in [12] for routing and link selection in networks comprising 100 nodes.

Furthermore, studies in MARL have demonstrated the efficacy of PPO in multi-agent scenarios, often outperforming other algorithms and/or achieving state-of-the-art performance in both collaborative and adversarial settings [10, 13]. While both CTDE and decentralized PPO formats have been explored, the latter has shown suboptimal performance compared to alternatives like QMix in specific environments such as SMAC [10]. Moreover, leveraging deep reinforcement learning in multi-agent settings with visual inputs has been investigated in [14]. Recently, multi-agent PPO (MAPPO) has also been applied to cooperative UAV trajectory design [15].

This research focuses on the specific problem of planetary exploration, using multi-agent PPO to find control policies in exploratory environments. Moreover, and specific to this work, several reward functions, environments, and swarm sizes are examined to foster the finding of exploratory policies that have emergent behaviors and generalize to unseen environments and swarm sizes.

III. Problem Formulation

This section describes how the MARL problem is formulated; first discussing where the policies obtained through the MARL algorithm can be deployed in a planetary mission, and later defining the formal optimization problem that needs to be solved for this end.

A. MARL Within the Planetary Exploratory Mission

In this study, a focus has been placed on contributing to areas with low TRLs [2], in particular; exploration, mapping and sampling, and cooperative task recognition and allocation. In this manner, a swarm exploration traceability model is proposed (see Fig.1), following NASA's *Methodology Fragment for Analysing Complex Multi-agent Systems* (MaCMAS) [16]. MaCMAS generates models at various abstraction levels to manage system complexity incrementally. This approach connects micro and macro level models, ensuring property verification across different system levels using formal methods. Additionally, MaCMAS offers techniques to refine and abstract models for comprehensive layer completion. Using MaCMAS for the mission design provides a clear reference of the area of contribution in this work, also highlighting other areas where research can be done.

In this manner, this study does not produce an exhaustive development of such a MaCMAS system, instead, it uses it to provide a starting point for envisioning a swarm planetary mission and contributing to it in a place with a low TRL. In the traceability model in Fig.1, six abstraction levels are envisioned; ranging from top-level mission planning, to agent-level control. Within each level, tasks/roles are defined; establishing a mission plan, identifying the task(s) that need to be executed, assembling a team of robots to do so, etc.

The MARL algorithm in this work aims at contributing to specific mission configurations, connected with black arrows in Fig.1. In such configurations, the mission entails using a swarm of land-based or airborne robots, all of the same type, who explore Points of Interest (POIs, which are also referred to as targets), and where the robots in the swarm can sense each other, but not directly communicate. Particularly, this work proposes to deploy MARL at the cooperation strategies level, a level of abstraction above agent-level control (see Fig.1), and for which there is a 3-5 TRL [2]. This way, the commands from the policy to the robot are guidance actions, stating where should the robot go next, based on its local perception. With these mission configurations, the swarms could explore complex terrains such as caves or lava tubes.

Here, the focus is on contributing to the following NASA requirements [2] (that is, not necessarily having complete fulfillment of all requirements, but showing that through MARL and the discovered policies, contributions are made with respect these requirements):

Req 1. Exploration, mapping, and sampling

1) **Autonomy**: the platforms should be able to operate for hours to days with no humans in the loop.
2) Relative pose estimation.
3) Formation keeping: the vehicles should be able to assess and, in many applications, control their relative location to a high degree of accuracy.
4) **Distributed estimation and cooperative mapping** to build real-time maps of the observed quantities and enable adaptive sampling strategies.

5) Distributed inter-agent communication.

Req 2. Cooperative Task and Task Allocation

1) **Recognise tasks** that should be performed based on environmental cues observed by the agents.
2) Assign such tasks according to the **agents' states and capabilities**.

As mentioned, from a mission perspective, **the goal is to obtain a control policy that can be used for the guidance command of robots in swarms performing planetary exploration.** This policy needs to be obtained using reward functions that follow straightforwardly from the mission requirements and require as little heuristics or human input as possible. In this way, scientists can potentially find policies that surpass human design or provide new insights into the control problem.

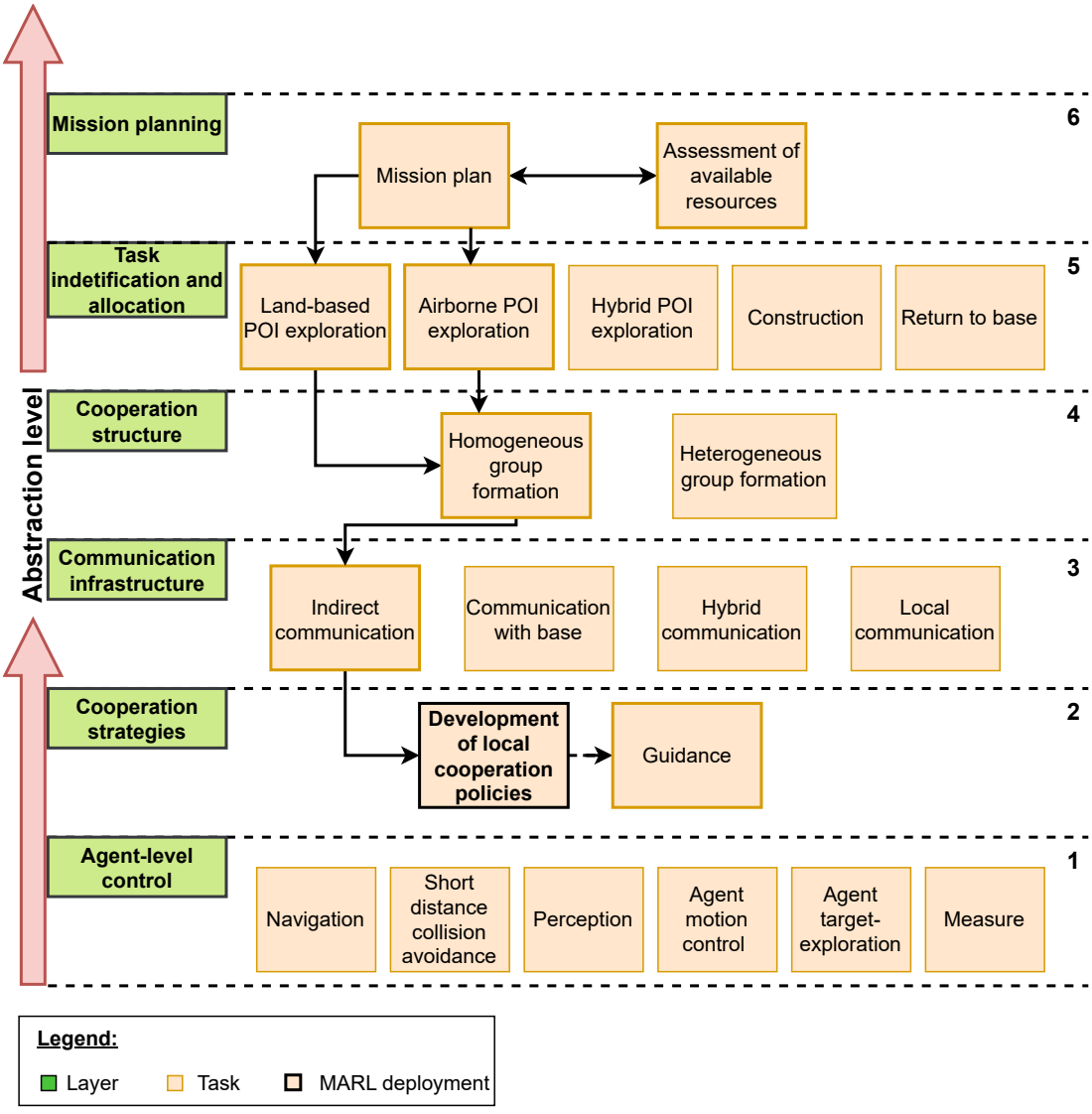

Fig. 1 Proposed MARL traceability model within NASA's MaCMAS methodology [16].

B. MARL Optimisation Problem

The collaborative planetary exploration problem is phrased as a study of decentralized partially observable Markov decision processes (DEC-POMDP)[7] with reward functions shared across all agents (which can depend on global or local performance, or a combination of both).
Furthermore, this study deals with homogeneous agent populations (see Fig. 1, abstraction level 4) where agents have similar observation and action spaces, and act according to the same policy π. We do not make restrictive assumptions for homogeneous agent populations, hence allowing for future extensions with heterogeneous agent swarms.

In this problem formulation, denoted by $\langle S, A, O, R, P, N, \gamma \rangle$, the state space S encapsulates the possible states of the planetary environment. A is the shared action space for each agent i (there is a total of N agents). Each agent's policy local observation o_i is represented as $O(s; i)$, where s is the global environment state. The shared reward function can take the form $R(s, A)$, $R(s_i, a_i)$, or $R(s_i, s, a_i, A)$; accounting for global or local performance, or a combination of both. The transition dynamics of the environment are captured by $P(s'|s, A)$, denoting the probability of transitioning from state s to s' given the joint action $A = (a_1, \ldots, a_n)$. The discount factor γ weighs future rewards against immediate ones. Agents deploy policies $\pi_\theta(a_i|o_i)$, parameterized by θ, to generate actions a_i based on their local observations. **The fundamental objective from the MARL perspective is to jointly optimize the discounted accumulated reward:**

$$J(\theta) = \mathbb{E}\left[\sum_{t=0}^{T}\sum_{i=1}^{N}\gamma^t R_t^i\right] \tag{1}$$

IV. Exploration Environment

In this work, a physically simulated two-dimensional exploration environment in continuous space and discrete time is used. This environment consists of N agents, M target locations (which simulate strategic locations that need to be explored during the mission), and Z obstacles that should be avoided. All agents, targets and obstacles inhabit a physical location in space and possess physical characteristics such as size. Once a target is explored it disappears from the environment. Also, agents take actions **simultaneously**.
Specifically, agents take guidance actions; stating the desired $\Delta x, \Delta y$ (change in horizontal and vertical position) to which to move next, but are also affected by interactions with the environment which, if they occur, result in a perturbed $\Delta x, \Delta y \to 0, 0$ (if the agents are taking an action which will move them within the bounds of an obstacle, their change in position is overwritten to not change, hence preventing agents from going through obstacles). The maximum allowed change in horizontal and vertical position is set to two distance units, and the action space consists of continuous actions. The transition dynamics for an agent i are shown below:

$$\mathbf{s_t^{(i)}} = \begin{bmatrix} x \\ y \end{bmatrix}_t^{(i)} = \begin{bmatrix} x + \Delta x \\ y + \Delta y \end{bmatrix}_{t-1}^{(i)} \text{ or } \begin{bmatrix} x \\ y \end{bmatrix}_{t-1}^{(i)} \text{ if an obstacle collision occurs} \tag{2}$$

In addition, each agent receives a reward signal which can solely be based on local or global performance, or a combination of both. Agents are rewarded when a target is found, and penalised for colliding with other agents or obstacles. Notice that unlike other works in MARL [9], we **do not** make the reward function dense by quantifying the norm distance to the nearest target, or similar heuristics. Although using such heuristics can enhance the learning convergence, it hinders the finding of policies that might outperform human-based knowledge, which is one of the final goals of using MARL for policy discovery.
The environment is terminated when all the targets are found, or a maximum number of timesteps is reached. Lastly, we use environments that have noticeably more complexity than the state-of-the-art in MARL planetary exploration [9] regarding the amount and placement of targets, agents, obstacles, and the nature of the reward functions. Fig.2 shows such an environment.

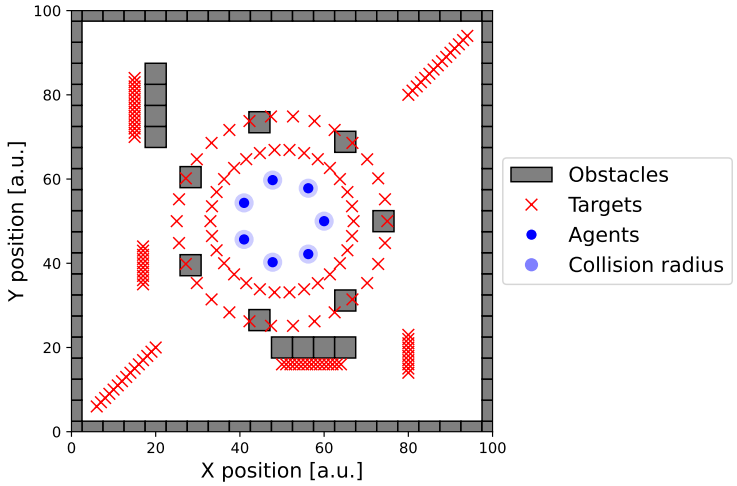

Fig. 2 Example exploration environment with seven agents (the circles), targets (the x's), and obstacles (the rectangles). Notice that targets are sometimes placed on top of obstacles, and hence can not be reached by the agents.

V. Learning Algorithm

To solve the distributed swarm control problem, a multi-agent extension of the PPO algorithm is proposed (see Algorithm 1) which is a variant of the algorithm used in [10], tailored for the collaborative exploration environment discussed in section IV, and including features to mitigate the MARL challenges listed in subsection II.A; discussed in this section and section VI.

A schematic of the learning process is shown in Fig.3. In this context, the agents in the swarm interact with the environment and collect experiences. These experiences are then used to learn a policy π_θ and a value function $V_\phi(s)$; two separate neural networks model these functions. These neural networks are shared across agents here since homogeneous swarms are studied, and one single buffer collects the experiences of all agents each time-step, effectively using **experience-based parallelization**. This technique enhances the sample efficiency of the algorithm as well as its performance [10]. However, the algorithm can be extended to heterogeneous agent settings by rolling different data buffers D, actor and critic networks per agent type, and generating agent-type-specific objectives and loss functions. Furthermore, the value function is only used during training (that is, once training is done, only the policy neural network needs to be implanted in the real robots) and is deployed for variance reduction.

As shown in Algorithm 1, the objective of the policy is to maximize the objective function $J(\theta)$, also shown in Eq.(3):

$$J(\theta) = \frac{1}{|D_k|T_\tau} \sum_{\tau \in D_k} \sum_{t=0}^{T_\tau} \min\left(\frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{old}}(a_t|o_t)} \hat{A}^{\pi_{\theta_k}}(O_t, a_t), g(\epsilon, \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{old}}(a_t|o_t)}) \hat{A}^{\pi_{\theta_k}}(O_t, a_t) \right) \tag{3}$$

In particular, the first term inside the `min` function operates on the ratio between the probabilities of taking an action under the new policy $\pi_\theta(a_t|o_t)$ and the old policy $\pi_{\theta_{old}}(a_t|o_t)$, and scaling by the advantage estimate $\hat{A}^{\pi_{\theta_k}}(O_t, a_t)$ (which measures the advantage of taking a certain action compared to the average action, and is calculated using the critic). The second term inside the `min` function is the clip function g, which saturates the scaling of the policy ratio by a factor $1 \pm \epsilon$.

This way, the `min` function computes the minimum of two values: the policy-scaled advantage estimate and the clipped version. If the behavior of the new policy is very different from the previous policy, the `min` function selects the clipped term, thus not promoting such an abrupt policy change.

This constraint helps PPO balance policy improvement with training stability.

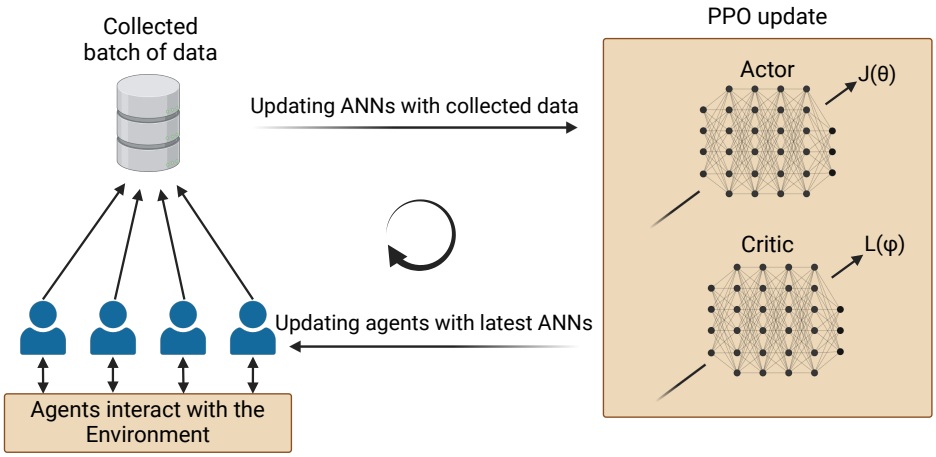

Fig. 3 Schematic of the multi-agent PPO learning process.

Furthermore, in this work the policy neural networks output deterministic actions (during deployment, the policies don't describe probability distributions). This is done to find a deterministic policy that can potentially be certified in future planetary applications. However, to make the algorithm explore during training, the outputs of the policy parameterize the mean vector of a multivariate Gaussian distribution with a predefined covariance matrix from which to sample actions. This exploratory noise is denoted as $\mathcal{N}$ in Algorithm 1. Notice that exploration techniques such as using entropy in the loss function are thus not implemented in this scenario.

Similar to the policy function, the goal of the value function is to minimise the loss function $L(\phi)$ [see Eq.(4)]. In this sense, a converged critic will have a good estimation of the value of being in a certain state, given the current policy. In multi-agent settings, this is a complex task, given that the environment is non-stationary, and having an accurate expectation of discounted rewards is thus difficult. Selecting an optimal observation of the state for the value function is crucial for obtaining robust learning, and section VI delves into the proposed implementation of the paradigm of Centralized Training with Decentralized Execution.

$$L(\phi) = \frac{1}{|D_k|T_\tau} \sum_{\tau \in D_k} \sum_{t=0}^{T_\tau} \left(V_\phi(O_t) - \hat{R}_t\right)^2 \tag{4}$$

Moreover, the neural networks used for the actor and critic (a feed-forward and CNN network, respectively) are updated using the Adam optimiser, since it is suitable for objectives that are non-stationary and for problems characterized by highly noisy and/or sparse gradients [17].
For reproducibility purposes, several clarifications can be made in Algorithm 1:

- All the hyperparameters (including the ones not listed here), can be found in Table 1.
- The batch size is defined to contain a maximum certain amount of experiences collected from the environment. The experiences τ collected by the agents are stored in the data buffer D_k. Once filled, D_k is used to update the actor and critic networks. After this, a new, empty buffer is initialised for the next k iteration.
- In $J(\theta)$ and $L(\phi)$ (the objective and loss of the actor and critic, respectively) the factor $|D_k|T_\tau$ refers to the size of D_k multiplied by the amount of maximum steps collected in a given environment episode. Since the environment can have a variable maximum number of steps T depending on the performance of the agents, this variability is indicated with T_τ.
- In this work, the advantage is calculated first with $\hat{A}_t = \hat{R}_t - V_\phi(O_t)$, and then z-score normalised with $\hat{A} \leftarrow \frac{\hat{A}-\mu_{\hat{A}}}{\sigma_{\hat{A}}}$, where the statistical properties of the advantages are calculated using the batch.
- In $J(\theta)$, g is the clip function.
- The perception (observations) of the actor and critic networks are differentiated with o and O. In the individual PPO configuration, the actor and critic receive the same observation $O = o$.

Algorithm 1 MAPPO algorithm for swarm exploration.

Initialise policy parameters θ, and initial value function parameters ϕ ▷ Initialise actor and critic
Set learning rate α, discount factor γ, updates per batch n, etc ▷ Initialise hyperparameters
while $k < k_{max}$ **do** ▷ Train for a set amount of steps
  Set data buffer $D_k = \{\}$ ▷ Batch of all agents data
  while $t <$ timesteps per batch **do** ▷ Collect environment experiences
    while environment *not done* **do**
      for all agents i **do**
        Collect actor and critic observations (o_t^i, O_t^i), actions $a_t^i = \pi_{\theta_k}(o_t^i) + \mathcal{N}_t$,
        action log probabilities $\log \pi_{\theta_k}(a_t^i|o_t^i)$, and reward R_t^i
        $\tau_t^i = [o_t^i, O_t^i, a_t^i, \log \pi_{\theta_k}(a_t^i|o_t^i), R_t^i]$
        $D_k += \tau_t^i$ ▷ Store experience in data buffer
      end for
      Act on the environment
    end while
  end while
  Compute rewards-to-go $\hat{R}_t$ on all experiences in D_k
  Compute z-score normalised advantage estimates, $\hat{A}_t$ (using any method of
  advantage estimation) based on the current value function V_{ϕ_k}
  Calculate actor PPO-clip objective:

$$J(\theta) = \frac{1}{|D_k|T_\tau} \sum_{\tau \in D_k} \sum_{t=0}^{T_\tau} \min\left(\frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{old}}(a_t|o_t)} \hat{A}^{\pi_{\theta_k}}(O_t, a_t), g(\epsilon, \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{old}}(a_t|o_t)}) \hat{A}^{\pi_{\theta_k}}(O_t, a_t) \right) \tag{5}$$

  Calculate critic loss:

$$L(\phi) = \frac{1}{|D_k|T_\tau} \sum_{\tau \in D_k} \sum_{t=0}^{T_\tau} \left(V_\phi(O_t) - \hat{R}_t \right)^2 \tag{6}$$

  for n steps **do** ▷ Update actor and critic networks
    Adam update θ on $J(\theta)$ with data D_k
    Adam update ϕ on $L(\phi)$ with data D_k
  end for
end while

VI. Perception Models

The selection of the states perceived by the policy and value functions is of paramount importance, as they must allow solving the DEC-POMDP, and also apply to a future potential deployment in an exploratory robot (this last requirement is specific to the policy). In this manner, both the policy and value (actor and critic) networks must have sufficient information to learn successful control strategies and value functions.

It should be noted that in this study, when centralized training with decentralized execution (CTDE) is used, the policy and value functions have access to different environment information, in other words, **they have different perceptions**; the critic receives some form of global information, whereas the actor is limited to perceive local information. This work uses the same nomenclature as [10]: the CTDE configuration for the multi-agent PPO is referred to as MAPPO (Multi-agent PPO), whereas not having CTDE is referred to as IPPO (Individual PPO).

In both IPPO and MAPPO, the actor uses a LIDAR-like perception model (the critic only uses this configuration in IPPO). A schematic of the LIDAR model is shown in Fig.4. To simulate the LIDAR, beams evenly "irradiate" from the agent, reaching a defined perception radius. Each beam has three channels detecting the closest agent, target, and obstacle, outputting the perception radius when nothing is detected. Distances are normalised for the policy's feed-forward neural network that maps LIDAR inputs to guidance actions.

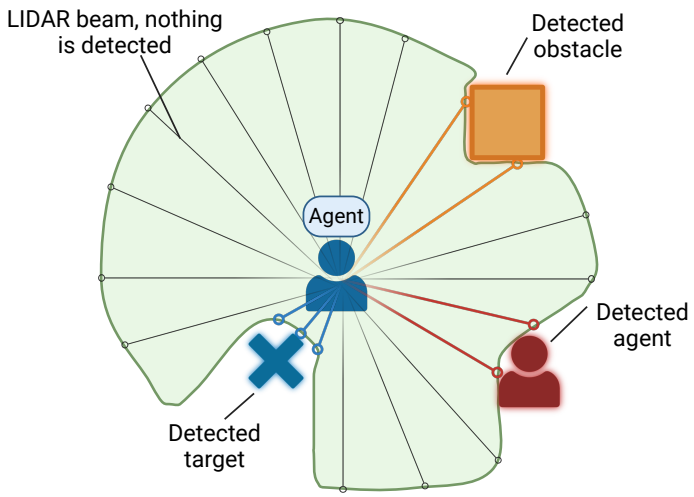

Fig. 4 Schematic of the LIDAR perception model used by the actor in both IPPO and MAPPO, and the critic in IPPO configuration.

In MAPPO configuration, the critic perceives a global environment tensor that also includes agent-specific information (see Fig.5). This tensor has three entries; entry i corresponds to the feature being extracted from the environment, and entries j and k to the horizontal and vertical coordinates of that feature space. In particular, four features are extracted from the environment: the position of all agents, targets, obstacles, and the self-position of the agent.

This way, the environment is discretised in a $M \times N$ grid where, for each environment feature, the grid cells in the tensor have a value of one if that feature is present, and zero otherwise. To process this input and model the critic, a deep convolutional neural network (CNN) is used, which estimates the value of the state defined by the input tensor.

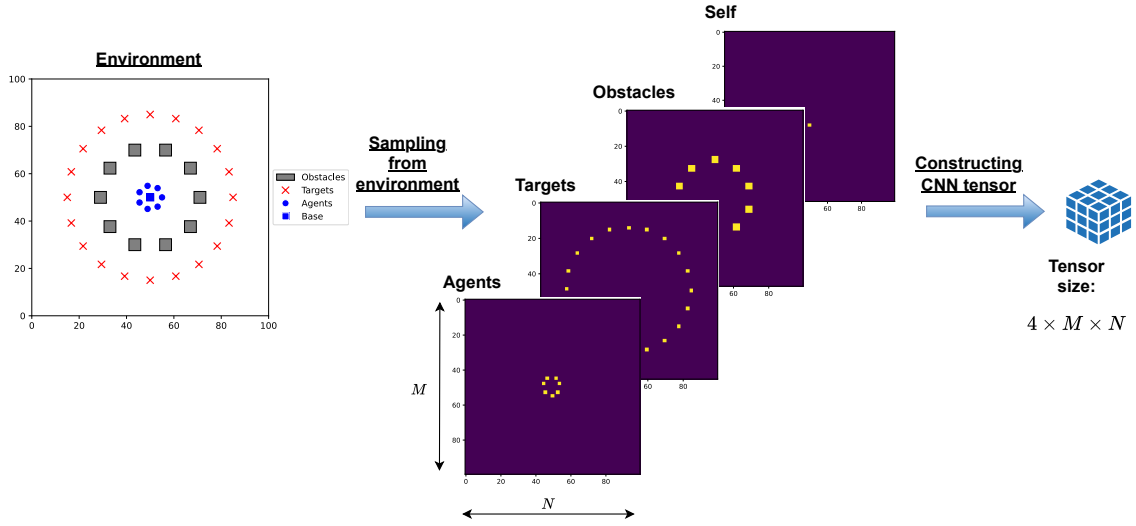

Fig. 5 Schematic of the generation of the MAPPO critic CNN perception tensor from the environment information. The tensor contains both global and agent-specific information (the self-position).

VII. Results and Discussion

This section presents the results of the tests carried out to investigate the proposed algorithm's performance, both during training and execution. In particular, the experiments allow a comparison between IPPO and MAPPO when learning using different types of reward functions, environment complexity, number of agents, and finally analyze the generalization/scalability capabilities of the learned policies.

All of these tests were performed using NVIDIA V100 GPUs, using the hyperparameters shown in Table 1. These parameters are obtained from [10] or else are experimentally found to be effective.

Moreover, these tests are computationally expensive, and due to computing resource constraints, one learning run has been performed per comparison setting. Hence, this work cannot fully address the statistical significance of the individual learning runs.

Table 1 Default hyperparameters.

Parameter	Value	Parameter	Value
Action noise σ^2	0.5 [a.u.]^2	Clip value	0.2
Collision Radius	2.1 [a.u.]	CNN pixel spatial resolution	100×100
Discount Factor (γ)	0.99	LIDAR perception radius	20 [a.u.]
Learning Rate (α)	0.005	Map Size	100×100 [a.u.]
Max allowed action	± 2 [a.u.]	Max Gradient Norm	10.0
Max Time-steps per Episode	100	Number of Agents	7
Number of LIDAR beams	36	Number of Obstacles	15
Number of Targets	140	Obstacle Height	5.0 [a.u.]
Obstacle Width	5.0 [a.u.]	Agent Spawn Radius (from map's centre)	10.0 [a.u.]
Time-steps per Batch	4800	Updates per batch (n)	5

A. Reward Function Types

In this experiment, three types of reward functions are compared; local, mixed, and global. This is shown in Eq.(7). Here, R_i refers to the reward received by agent i, C_i is the collision status of the agent, and T refers to whether an agent has found a target in the current step. Lastly, j refers to a specific agent within a swarm containing N agents.

$$\begin{aligned} &\textbf{Local Reward} && \textbf{Mixed Reward} && \textbf{Global Reward} \\ &R_i = -\frac{C_i}{7} + T_i && R_i = -\frac{C_i}{7} + T_i + 0.2 \cdot \sum_{j=1}^{N} T_j && R_i = -\frac{C_i}{7} + \sum_{j=1}^{N} T_j \end{aligned} \tag{7}$$

Hence, the local reward function solely depends on the individual performance of agent i. The mixed reward function is similar but also rewards when the swarm finds targets, although with lower relevance than individual exploration. Lastly, the global reward function sanctions agent-specific collisions and rewards the finding of targets by the overall swarm.

Moreover, different scaling for collision penalties were studied, however, the scaling (7^{-1}) of the collision penalty is smaller than the weight of finding a new target and is always linked to individual performance since it was found that otherwise, the agents fall in the local minima of not moving.

Importantly, these reward functions have different scaling for a similar swarm performance. This is caused by the summation terms in the mixed and global reward functions. Due to this reason, the learning curves across different reward functions are not compared in terms of performance, but the learned policies are.

The learning progress of IPPO and MAPPO is shown in Fig.6. Several observations can be made. Initially, all algorithms experience a rapid improvement in the rewards obtained, and the learning performance flattens afterward. This behavior agrees with similar results from other works [10].
Secondly, the training performance of IPPO and MAPPO is comparable in the local and mixed settings, while MAPPO doubles the performance of IPPO in global reward settings.
With local rewards, IPPO achieves better performance (close to episode $0.4 \cdot 10^5$), although it is possible that with longer run times or different initialisations, MAPPO reaches a similar performance.

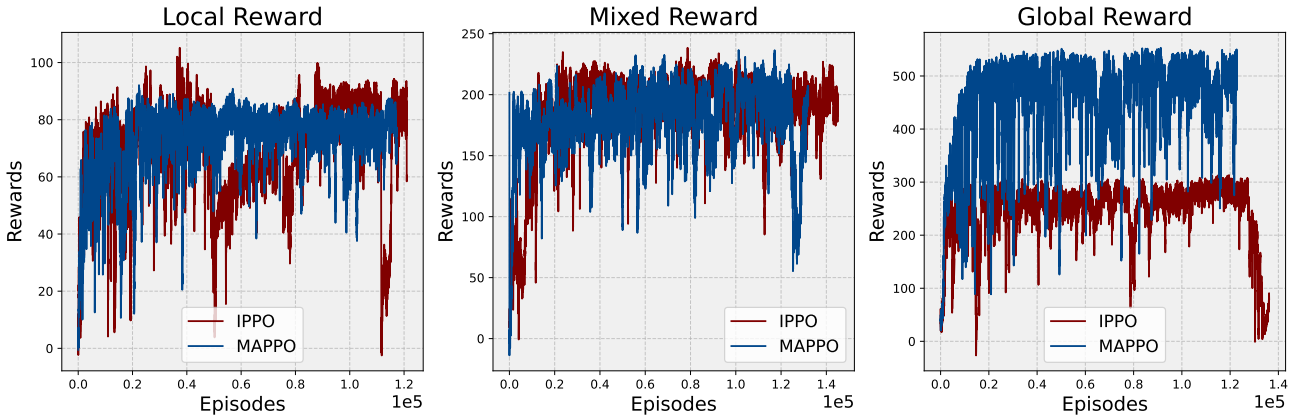

Fig. 6 Learning performance of the IPPO and MAPPO models with the respective type of reward function.

Moreover, when looking at the training stability, the variances of the learning curves are shown in Fig.7. Here, it is observed that for local rewards, the variance of MAPPO is always similar or significantly smaller than IPPO's. With mixed rewards, IPPO and MAPPO have comparable variances, with IPPO achieving marginal higher learning stability. Then, MAPPO has significantly higher variances than IPPO with global rewards (also occurring by the fact that it is achieving higher rewards), although after episode $1.2 \cdot 10^5$, IPPO has a catastrophic training instability; decreasing the obtained reward to be close to zero.
The exponential variance reduction behavior shown in all scenarios is attributed to taking the variance of a non-stationary value, reflecting the transition from rapid to slower reward increases.

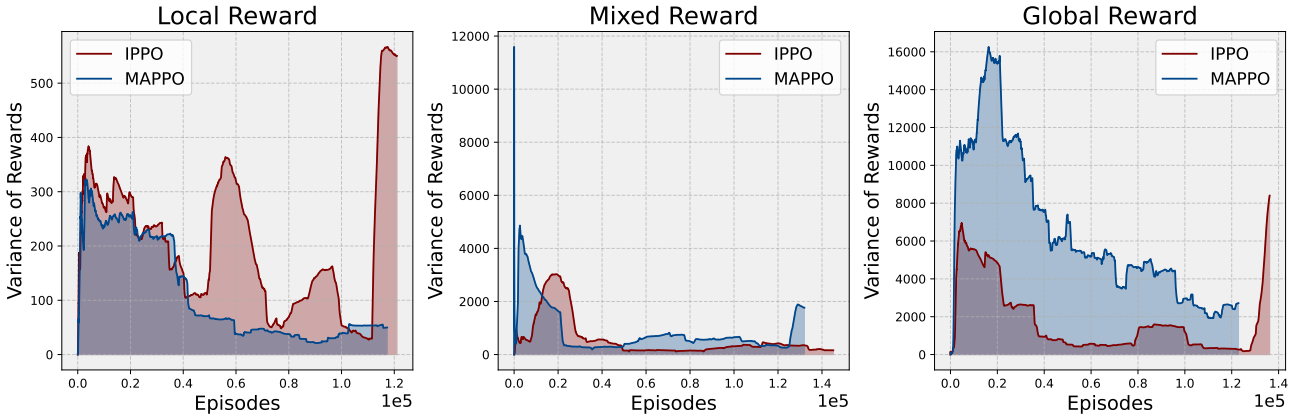

Fig. 7 Variance of the rewards obtained during training. The variance is calculated using a sliding window of 3,000 episodes.

From these results, it is observed that MAPPO and IPPO have similar learning performance when the rewards are not based (or weakly so) on global swarm performance and instead focus more on agent-specific metrics. Noticeably,

MAPPO doubles the amount of rewards obtained by IPPO when this is not the case and the rewards are global.

This attests to the importance of the state representation perceived by the actor and critic networks. Compared to the CNN tensor, the LIDAR perceives richer information near the agent, detecting nearby environment features accurately, unlike the CNN tensor, which is limited in resolution by the selected pixel size. Thus, when considering IPPO with local or mixed rewards, the critic perceives information that is highly relevant for estimating the current value, whereas the CNN network is limited in local perception, although it has access to global information that can be relevant during further training progress (not considered here) where complex egocentric behaviors can emerge to increase the local reward (such as strategically stealing targets that could be captured first by other agents) .

However, when the rewards become global, the learning behavior of IPPO becomes critically hindered, unlike MAPPO's, which can still achieve good learning performance. In this scenario, knowing global environment information plays a crucial role in estimating the value function, since agents need to identify that they can receive rewards based on the behavior of other agents, and not just them, and crucially, that through cooperative behaviors, their rewards can be maximised. On the contrary, when using a LIDAR-critic in this scenario (IPPO), the agent receives rewards according to information they can not perceive and is not stationary, thus making the learning problem extremely challenging. These findings agree with the theory behind CTDE, and the usage of these techniques in multi-agent settings [10, 13].

As a remark, the different reward functions change the nature of the problem. Local reward functions result in adversarial training, as the different agents in the swarm compete for a limited amount of targets. On the contrary, the global reward function fosters the development of cooperative policies, which is the end goal of this study. From this perspective, MAPPO is the preferred algorithm to deploy.

B. Rich Environment Training

To test whether training in random environments can enhance the performance of the learned policy (following the findings from [18]), a rich environment training is used to also compare against always training in the same environment.

In the rich environment, the targets and obstacles are generated randomly every time the environment is initialised (although some obstacles near the map's center are always generated in the same place), and the total amount of obstacles can change. Moreover, rich environments contain the same number of targets as those used in subsection VII.A. Lastly, although the agents are always initialized evenly spaced from the map's center, in a circular formation, the agents' initial position within the circle is generated randomly.

The three algorithms showing better performance in subsection VII.A are considered: IPPO with local rewards, and MAPPO using mixed and global rewards (in the mixed reward setting, IPPO and MAPPO displayed similar performance, but given its CTDE nature which should favor stronger results [10], MAPPO is selected). Fig.8 shows the learning progress of the three algorithms.

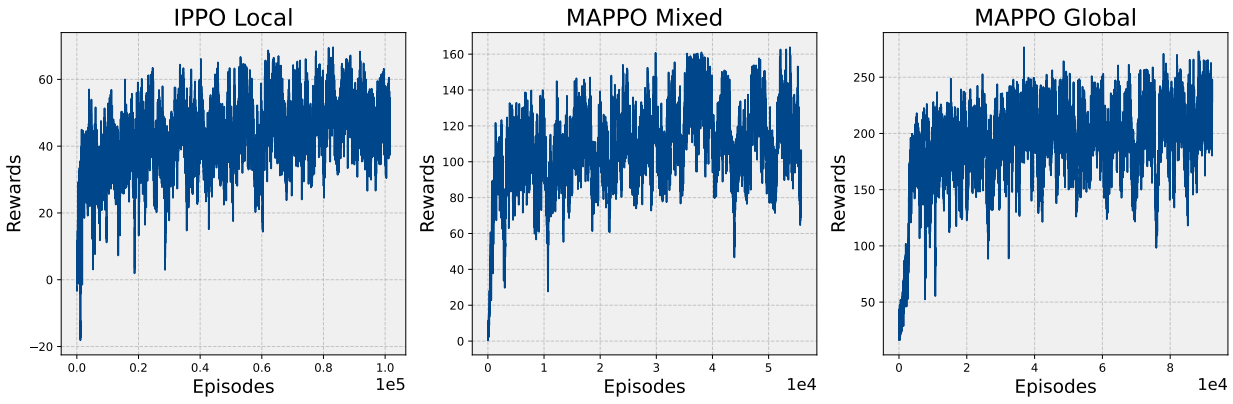

Fig. 8 Learning performance of the models in the rich environment. The models have been trained for different amounts of episodes due to constraints in computational resources.

Similar to the learning curves shown in subsection VII.A, the three algorithms experience a rapid increase of rewards

at the beginning of training, slowly improving during the rest of the run.

Moreover, all three algorithms achieve lower rewards than the ones obtained always training in the same environment (see Fig.6 for a comparison). Although this difference in performance might also be caused by the shorter training times (the MAPPO algorithms are roughly trained for half the time of those shown in Fig.6, due to computational resources constraints), this reduction in performance might further be accentuated by the learning problem becoming more difficult, as over-fitting to explore targets or avoid obstacles that are always expected to be located in the same place is not possible. It is hypothesized that agreeing with the findings from [18], with longer learning times the algorithm can potentially find more generalizable policies leading to higher rewards, as the rich environment has fewer properties that can be exploited with simple heuristics.
However, as shown in subsubsection VII.D.2, the learned policies exhibit different behaviors from the ones obtained learning always in the same environment, and generalize better (see subsection VII.E).

C. Training With a Different Amount of Agents

Training with three agents has been tested to compare the agent scalability of the learning algorithms. Global rewards are used to ensure a fully cooperative setting, and both IPPO and MAPPO are examined. Furthermore, the training environment is identical to the one used with seven agents in subsection VII.A.
Fig.9 shows the learning progress of both IPPO and MAPPO. Here, it is observed that both algorithms gradually improve the obtained return, with IPPO achieving higher average rewards. This difference in ultimate reward is caused by the IPPO agents finding targets in the top left of the map, something the MAPPO agents fail to do (see subsubsection VII.D.3). It is hypothesized that with different training initializations, MAPPO could achieve similar average performance.

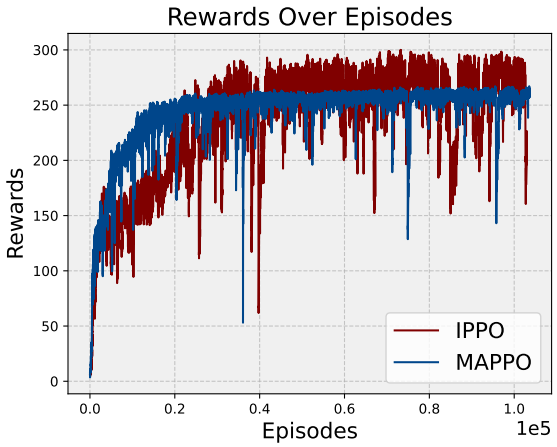

Fig. 9 Learning performance of IPPO and MAPPO in an environment with three agents, and global rewards.

A noticeable contrast lies in the training stability. The MAPPO training curve has a variance that is between two and three orders of magnitude smaller than IPPO's, as shown in Fig.10 (the initial high variances of both algorithms are caused by the rapid increase in rewards obtained at the beginning of the training phase, and due to its non-stationary behavior, the variances are **not** used to assess learning stability in this region). This finding further reflects the importance of the input to the critic in multi-agent settings, and how using CTDE stabilises the learning run.

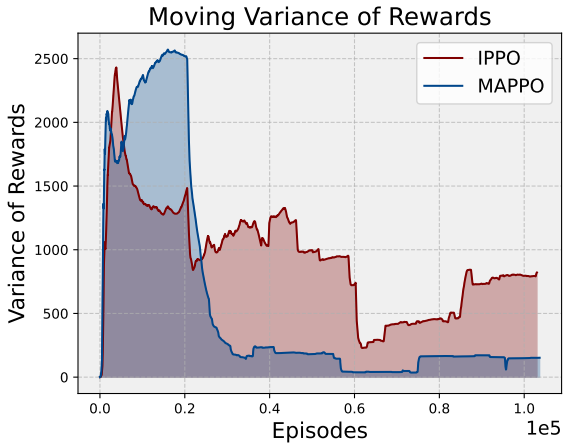

Fig. 10 Variances of the rewards obtained training with 3 agents (see Figure 9). The variance is calculated using a sliding window of 3,000 episodes.

Moreover, it is found that MAPPO has better agent scalability than IPPO when using global rewards. This is reflected by the fact that the MAPPO learning runs with seven agents achieve higher rewards than the ones with three agents (effectively doubling), while IPPO fails to improve (this is observed by comparing Fig.6 and Fig.9). Agreeing with the discussion in subsection VII.A, CTDE is thus likely to be an essential feature when developing MARL algorithms that can be used to learn policies in large swarms (when global reward functions are used).

Nevertheless, with both three and seven agents, the algorithms fall into suboptimal solutions (the maximum possible reward, related to finding all targets and not crashing, is never found). This suboptimal behavior is hypothesised to be caused by limited training episodes. Consistent with the theory discussed in subsection II.A, this reflects MARL algorithms' susceptibility to get trapped in local minima.

D. Policy Analysis

The behaviors of the learned policies are now studied; examining the policies obtained in subsection VII.A, subsection VII.B, and subsection VII.C.

1. Reward Functions

Inspecting the behaviors of IPPO and MAPPO with local rewards in Fig.11, IPPO explores more targets, with the pink agent exploring the bottom left targets too, something that the MAPPO agents fail to achieve. This could be caused by a more unfavorable training initialisation, and longer/more training runs would be needed to assess this.
However, it is observed that the behaviors of the agents are very different. The IPPO agents move in more erratic patterns, often following similar trajectories and hence stealing targets from each other, with MAPPO agents moving circularly to collect the targets near the center of the map (although it can be observed that the red MAPO agent in Fig.11 gets stuck between obstacles).

When it comes to mixed rewards, similar behaviors are observed (see Fig.12). In this case, both algorithms achieve a similar exploration of the environment, with MAPPO having a more collaborative exploration of the inner circle of targets. Also, both algorithms explore the lower-left targets with the pink agent and have agents that crash against obstacles.

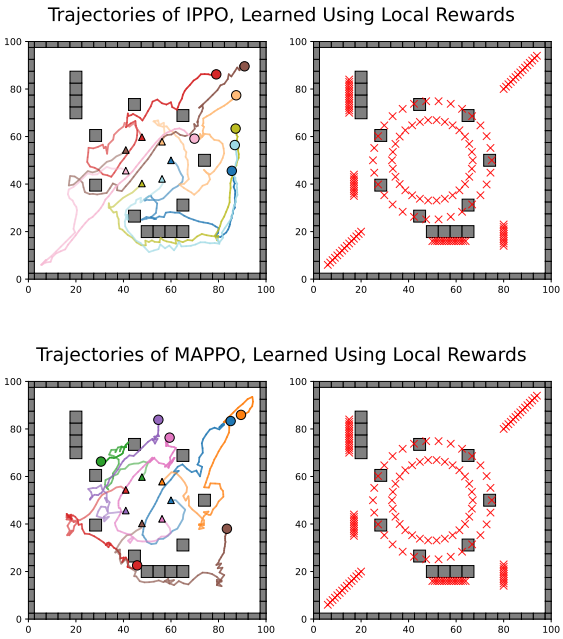

Fig. 11 IPPO (top) and MAPPO (bottom) policy behavior in the training environment, using local reward functions. Left: Trajectory traces of the agents, the triangles indicate the starting position of the agents, and the circles, their final positions. Each color corresponds to a specific agent. Right: environment drawing, only containing the location of the obstacles and targets. Here IPPO and MAPPO explore 115 and 97 targets, respectively. The environment has a total of 141 targets.

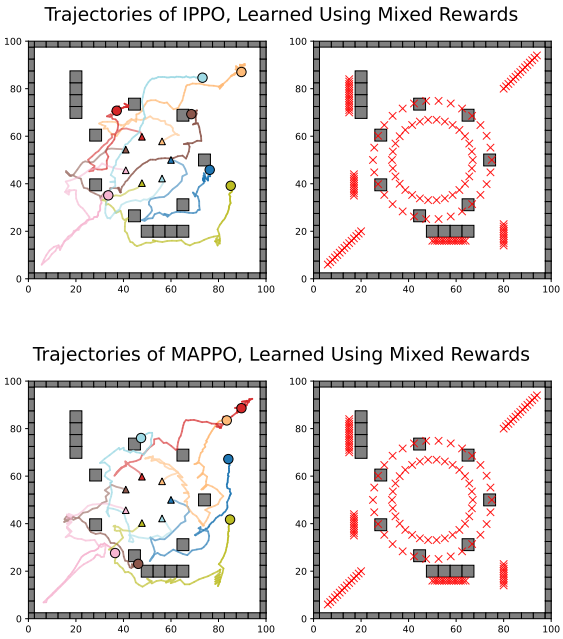

Fig. 12 IPPO (top) and MAPPO (bottom) policy behavior in the training environment, using mixed functions. Left: Trajectory traces of the agents, the triangles indicate the starting position of the agents, and the circles, their final positions. Each color corresponds to a specific agent. Right: environment drawing, only containing the location of the obstacles and targets. Here, IPPO and MAPPO explore 114 and 113 targets, respectively. The environment has a total of 141 targets.

Finally, Fig.13 shows the policy trajectories with global rewards. In both cases, the algorithms only find policies that explore a specific diagonal of the environment, with MAPPO having collaborative exploration of the majority of the inner targets' circle and better agent spread, also exploring the lower part of the map. The performance degradation of both algorithms when compared with the other reward settings is likely caused by the increased difficulty of using global rewards, and more training time might be needed to achieve performances similar to the local and mixed rewards counterparts.

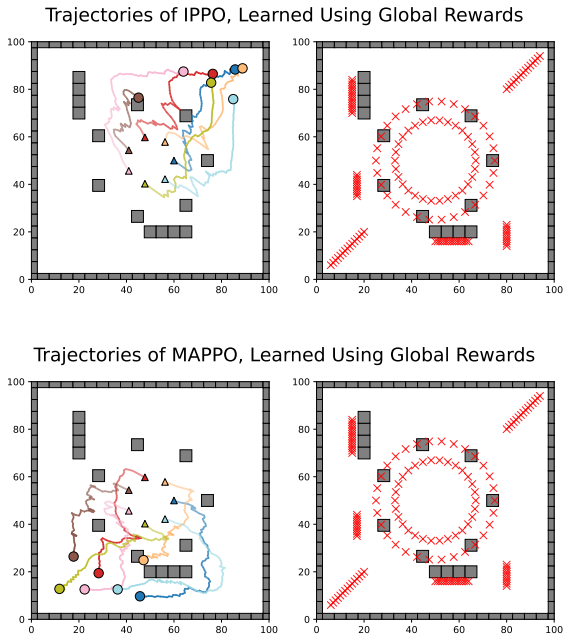

Fig. 13 IPPO (top) and MAPPO (bottom) policy behavior in the training environment, using global reward functions. Left: Trajectory traces of the agents, the triangles indicate the starting position of the agents, and the circles, their final positions. Each color corresponds to a specific agent. Right: environment drawing, only containing the location of the obstacles and targets. IPPO and MAPPO explore 47 and 82 targets, respectively. The environment has a total of 141 targets.

From these results, it can be observed that while there are no significant differences in performance between local and mixed rewards, the global reward scenario obtains policies that fail to explore as many targets. This performance difference is hypothesised to decrease with more training time, however, it highlights that for reduced training times, local or mixed rewards have the potential to learn higher performing policies, since the global rewards make the learning problem more complex.

2. Rich Environment

Inspecting the behaviors of IPPO and MAPPO learned in a rich environment (shown in Fig.14), it can be observed that the learned policies are very different from the ones obtained training in non-rich environments.

Firstly, both IPPO and MAPPO-mixed (using mixed rewards during training) agents learn a territorial behavior (see the top two plots of Fig.14), where they spread over the map and attempt to explore areas not covered by other agents, thus maximizing the probabilities of finding regions in the map with unexplored targets. This behavior might emerge because of the adversarial nature of the problem, imposed by the local and mixed reward functions. The policies also result in the agents sometimes colliding with obstacles. This could be caused by the limited training time and the scaling of the collision penalty; since the latter is much smaller than the one obtained when finding a target, it is plausible that the agents first prioritize the finding of targets, since this has a more salient effect on the reward function (for example, as shown in Eq.(7), for the local reward function, finding a target has seven times more relevance than colliding).

Regarding MAPPO training with global rewards, the agents learn to collectively sweep the area (see Fig.14), keeping within a distance of each other, and exploring targets as they move towards the lower right part of the environment. This is a learned collaborative and cooperative behavior, where the swarm agents work together on the shared goal of collecting targets (established by the global reward function) and cooperating in agreeing to stay a "safe" distance from each other, each agent allowing others to explore targets if they are closer to them.

An important observation can be made from this experiment. Compared with the global reward, the local and mixed rewards achieve better exploration of the environment. This highlights the fact (shown analogously in the previous section) that even though MAPPO performs better than IPPO when using global rewards, depending on the learning problem, using IPPO or MAPPO with a different type of reward function can achieve better performance. The emergent patterns obtained using MAPPO with global rewards are more pronounced, but this does not necessarily lead to an increase in swarm performance, at least with this amount of training.

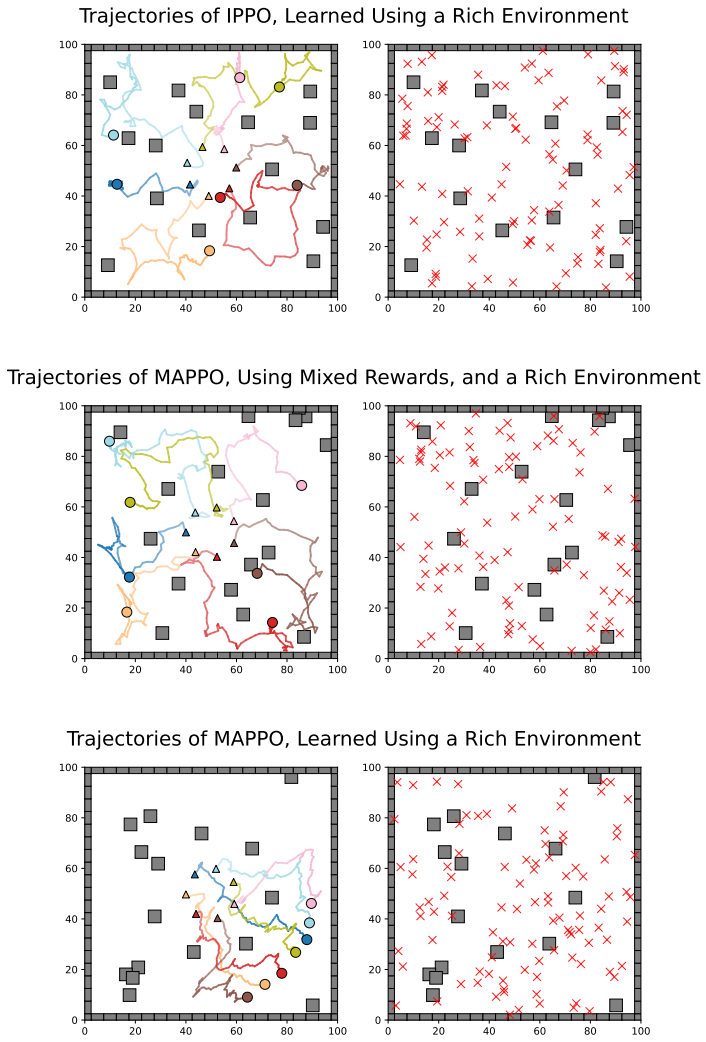

Fig. 14 IPPO-local (top), MAPPO-mixed (middle), and MAPPO-global (bottom) policy behavior in the rich training environment. Left: Trajectory traces of the agents, the triangles indicate the starting position of the agents, and the circles, their final positions. Each color corresponds to a specific agent. Right: environment drawing, only containing the location of the obstacles and targets. Both IPPO and MAPPO-mixed achieve a spreading behavior, while MAPPO agents learn to jointly sweep the map towards the bottom right part of it.

3. Training With 3 Agents

When training with 3 agents, two different patterns **emerge**, shown in Fig.15. The IPPO agents learn to explore the inner circle of targets collectively, each exploring one-third of it, and then proceed to explore targets in different areas of the environment, thus strategically not impeding each other.

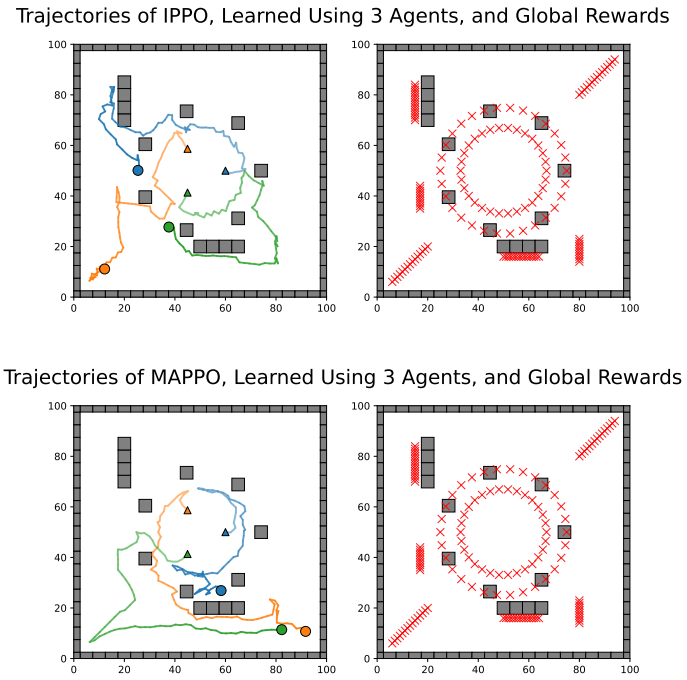

Fig. 15 IPPO (top) and MAPPO (bottom) policy behavior in the training environment, using global reward functions and 3 agents. Left: Trajectory traces of the agents, the triangles indicate the starting position of the agents, and the circles, their final positions. Each color corresponds to a specific agent. Right: environment drawing, only containing the location of the obstacles and targets. IPPO and MAPPO explore 107 and 91 targets, respectively. The environment has a total of 141 targets.

Moreover, the MAPPO agents learn a different exploration strategy. They also allocate tasks to different agents (see the lower plot in Fig.15), however, this is done differently. Inspecting the behavior of each MAPPO agent, shown in Fig.16, the blue agent gets assigned the exploration of the majority of targets in the inner circle, while the orange agent explores a small region of the inner circle and then focuses on exploring the lower part of the map, the green agent focusing on the cluster of targets in the lower-left part of the map.

These policies show **cooperative motion planning strategies, and task recognition and allocation behaviors**. Compared with the previous environment configurations, the emerging strategies are clearly defined. This could be caused by the decrease in complexity of the learning problem, as it goes from 7 to 3 agents. It is thus possible that if the algorithms are allowed to train for longer times, such defined patterns can emerge for 7 or more agents too. Furthermore, both IPPO and MAPPO achieve these strategies. Regarding IPPO, there is a notorious performance gain when compared to when it learns using seven agents and global rewards (see Fig.6). This might occur because the variance in the rewards received reduces with fewer agents. Furthermore, in the discovered emergent strategy of collaborative inner circle exploration, all agents remain within LIDAR reach of each other, thus in this case, the critic of IPPO perceives the necessary information to determine the value of the current state and foster the emergence of the circular motion pattern.

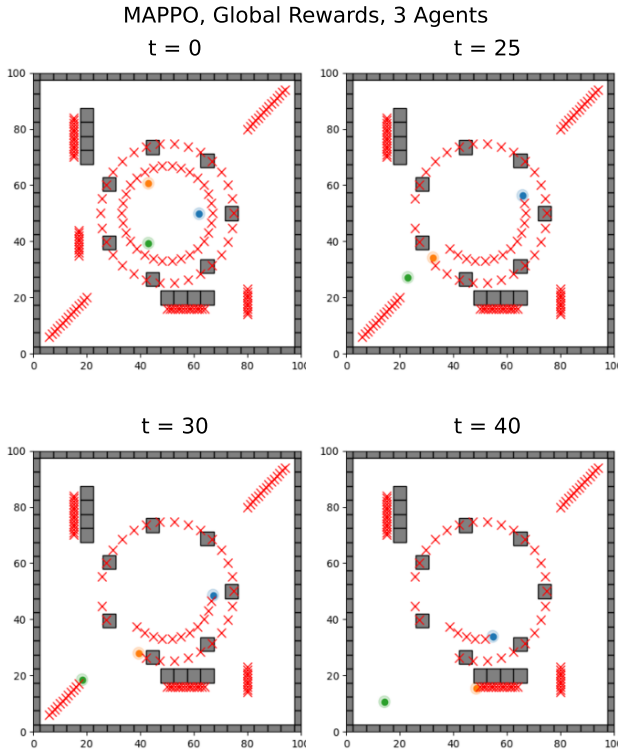

Fig. 16 Trajectory trace of the learned policy from MAPPO, using local reward functions. The targets disappear from the environment as they are explored. It is observed that the agents autonomously distribute the task; the blue agent explores the inner target circle, the green one explores the targets in the bottom-left of the map, and the orange explores the targets in the lower part of the environment.

E. Agent Scalability During Deployment

To further understand the applicability of the learned policy to swarm missions, the generalisation properties of the policy are studied. Specifically, policies from algorithms trained in rich versus non-rich environments are compared. In this experiment, the **learned** policies are deployed on swarms with different numbers of agents, to examine whether the behaviors learned training with seven agents can generalize to different swarm sizes. Each simulation run consists of 70 steps, and the environments generate 100 target positions. Moreover, three metrics are examined: the amount of explored targets, and the collisions with other agents and obstacles. Regarding the collision count, the total amount of collisions during the simulation is calculated by adding together all the individual collisions of the agents. In the case of the collisions between agents, when two agents collide, this is counted as a single collision. However, if an agent gets stuck in a crashing behavior against an obstacle or fellow agent, that collision is counted at each time step (e.g., an agent colliding with the same obstacle for 100 time steps counts as 100 collisions). Lastly, the algorithms trained with a rich environment are referred to as "Rich Env".

Fig.17 shows IPPO scalability with local rewards. Larger swarms explore more targets. Also, IPPO Rich Env explores more targets than IPPO in all the different swarm sizes and has fewer agent-agent collisions. Regarding obstacle collisions, both algorithms have a similar performance; the number of collisions increases as the swarm gets larger. For IPPO Rich Env, target exploration saturates beyond 11 agents because the environment generates only 100 targets, some unreachable due to obstacle overlap. Thus, exploring close to 100 targets represents maximum performance.

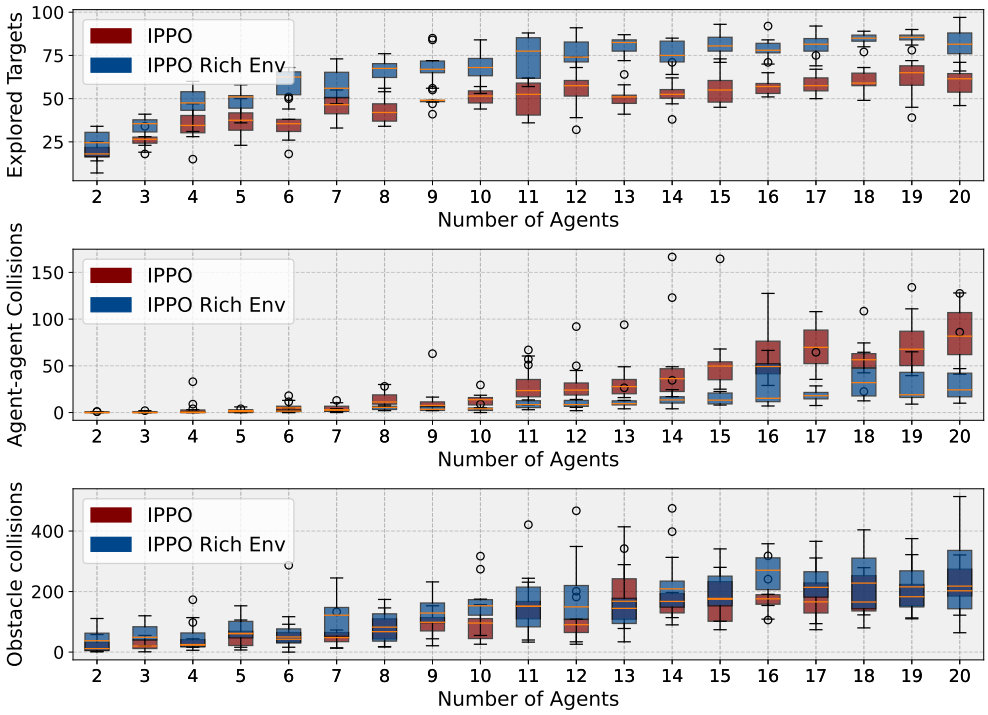

Fig. 17 Agent scalability plot of IPPO training with and without a rich environment. Local rewards are used during training. Lastly, the box plots are constructed using 10 runs per number of agents. The deployed policies are trained using 7 agents.

The scalability of MAPPO using mixed rewards is shown in Fig.18. Similar to the IPPO scenario, MAPPO Rich Env finds more targets than MAPPO for all numbers of agents and has fewer agent-agent collisions (although this last difference in performance is marginal). Moreover, both algorithms have a similar number of obstacle collisions, with MAPPO having marginally fewer collisions.

When inspecting MAPPO with global rewards in Fig.19, both MAPPO and MAPPO Rich Env explore more targets as the swarm gets larger, but there is no significant performance difference between them. It should be noted that both algorithms have an asymptotic scalability behavior: exploring a maximum of 45 targets (approximately). Since the environments are generated with 100 targets, this behavior is not caused by exploring all available targets, but rather due to the policys' failure to generalize enough to use all the swarm agents effectively.

Regarding the number of agent-agent collisions, MAPPO Rich Env has fewer collisions, its collisions scaling approximately linearly with the number of agents, while MAPPO has a weak exponential behavior. Moreover, MAPPO has more obstacle collisions than MAPPO Rich Env.

Lastly, when comparing across types of reward functions (that is, Figs.17, 18, and 19), several observations can be made:

- All algorithms find more targets as the swarm gets larger. However, while IPPO and MAPPO-mixed (and their Rich Env versions) effectively explore the total amount of targets in the map, MAPPO-global fails to do so, having close to a 50% performance degradation. This is not expected and could be caused by the harder nature of using global rewards, thus needing more training time until efficient exploratory behaviors emerge.
- Regarding agent-agent collisions, the rich environment algorithms achieve similar performances, with MAPPO-mixed and MAPPO-global Rich Envs outperforming IPPO. Regarding the non-rich environment algorithms, IPPO and MAPPO-global show weak exponential scalability, thus being noticeably worse than MAPPO-mixed, which shows linear scalability. In this sense, MAPPO has better performance than IPPO. This is expected since using CTDE can foster the emergence of policies that result in coordinated behaviors.
- Concerning obstacle collisions, the rich environment algorithms collide less, with the exception of MAPPO-mixed. This could occur because the training time in the rich environment is half that of the one done in the non-rich case, and with further training, it is theorized that the performances can be similar.

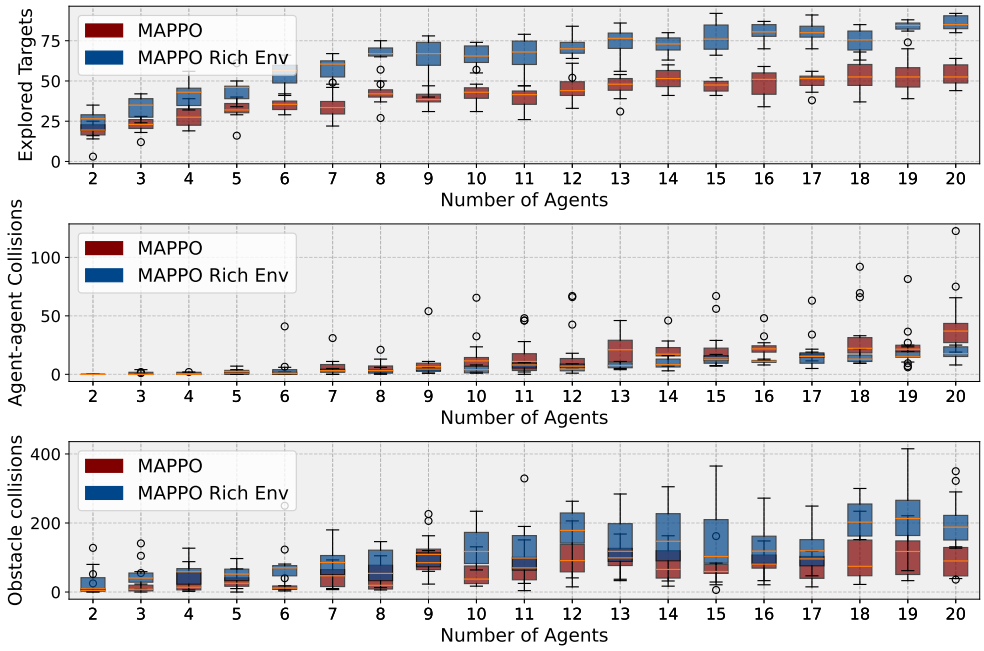

Fig. 18 Agent scalability plot of MAPPO training with and without a rich environment. Mixed rewards are used during training. Lastly, the box plots are constructed using 10 runs per number of agents. The deployed policies are trained using 7 agents.

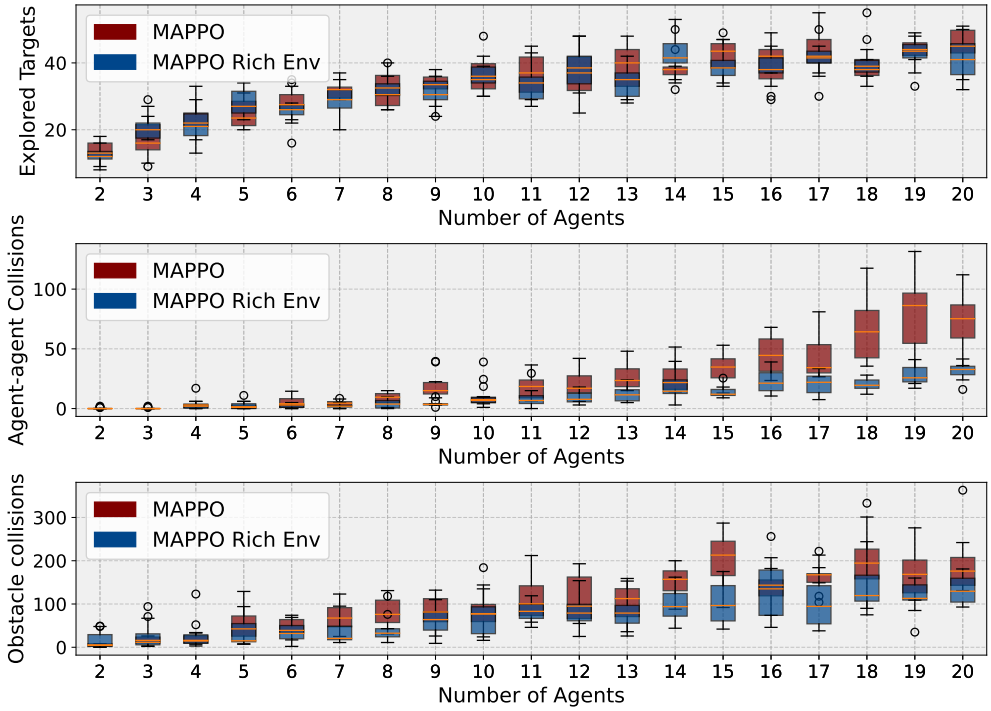

Fig. 19 Agent scalability plot of MAPPO training with and without a rich environment. Global rewards are used during training. Lastly, the box plots are constructed using 10 runs per number of agents. The deployed policies are trained using 7 agents.

- In all scenarios, **training with a rich environment achieves similar or better performance than training without. This is a noticeable performance gain, given that the Rich Env algorithms trained for less time**, and aligns with the findings of [18], although this work extends it to a multi-agent scenario.
- From the aforementioned observations, **MAPPO-mixed Rich Env achieves the best scalability performance**.

F. Improving the TRL of Swarm Missions

After examining the learning performance of IPPO and MAPPO with different reward functions and environments, it is important to assess whether the policies contribute towards the NASA requirements listed in section III. In particular, the learned policies contribute to the following requirements:

Contributing to Req 1. Exploration, mapping, and sampling

1) The agents can explore autonomously, and the algorithms have no limitations in operating times.
2) Relative pose estimation is not addressed by these results.
3) The agents can keep formations and achieve collaboration behaviors.
4) The algorithm can be used as a first step to develop a distributed cooperation and mapping system, already showcasing adaptive sampling strategies.
5) The agents indirectly achieve inter-agent communication. However, a study of direct communication channels is not addressed in this work.

Contributing to Req 2. Cooperative Task and Task Allocation

1) The agents can recognize the task that needs to be performed based on LIDAR perception, hence considering environmental cues.
2) The agents dynamically allocate the exploration of targets based on the swarm state.

As an important remark, the algorithms fulfill these requirements to a certain extent, but **not fully**. However, it is hypothesized that with longer training runs, bigger artificial neural networks, higher fidelity simulations, and thorough policy verification and validation, this framework can eventually be deployed in real swarm systems.

VIII. Conclusion

This work combines two fields; swarm space exploration, and multi-agent reinforcement learning. This is done by phrasing the swarm learning problem as a decentralized partially observable Markov Decision Process and considering planetary mission requirements. Moreover, the learned policies are placed within a NASA MaCMAS architecture, taking guidance actions. This is done within a simulated exploration environment where agents have target locations they need to explore, and obstacles to avoid. This environment was phrased to have similarities to the previous state-of-the-art [9], but severely increasing the complexity of the learning problem regarding environment features, the use of sparse reward functions that can be obtained from mission requirements, and the number of agents.

To solve the learning problem, a multi-agent extension of the PPO algorithm has been developed, which employs LIDAR perception for the policy, and a global environment tensor for the CNN of the critic when in CTDE (referred to as MAPPO). In this sense, both the environment and the algorithm provide a further step toward learning policies that can be used in a real exploration mission.

To further study the nature of the problem, the IPPO and MAPPO configurations are compared with reward functions that only account for agent-specific behavior, full swarm behavior, or a combination of the two. The IPPO configuration of the algorithm, using a LIDAR for the critic, achieves comparable performance to MAPPO when using local or mixed reward functions but, notoriously, MAPPO doubles IPPO's performance when using global reward settings, and achieves more stable learning in most scenarios.
Moreover, both IPPO and MAPPO can learn with more agents than the algorithm in [9] (7 vs 5). Also, the algorithms learn policies that achieve emergent cooperation behaviors and attain good learning scalability; tested training with both 3 and 7 agents.

Furthermore, **depending on the nature of the cooperative learning problem**, global reward functions are found to unnecessarily complicate the learning task, as there are scenarios where using local or mixed rewards also obtains

high-performing cooperation policies (sometimes better than the ones obtained with global rewards).

Moreover, training in a rich environment achieves better generalization and scalability capabilities compared to always training in the same environment. This was achieved for all reward function types. This is a noticeable performance gain given that the rich environment algorithms trained for shorter times.

Additionally, for all tested algorithms, the learned behaviors scale to swarm sizes not seen during training, thus highlighting the potential of using MARL to learn policies that are flexible to different numbers of agents.

Finally, the learned policies improve the TRL of swarm planetary exploration missions, contributing to exploration, mapping and sampling, and cooperative task and task allocation, according to NASA requirements.

References

[1] Thai, Z. W., Balasubramani, P., Brand, C., Haines, A., and DeLaurentis, D. A., "Study of Swarm-based Planetary Exploration Architectures Using Agent-Based Modeling," *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, 2020. https://doi.org/10.2514/6.2020-0075, URL https://arc.aiaa.org/doi/10.2514/6.2020-0075.

[2] Rahmani, A., Bandyopadhyay, S., and Rossi, F., "Space Vehicle Swarm Exploration Missions: A Study of Key Enabling Technologies and Gaps," *70th International Astronautical Congress*, 2019.

[3] Vitug, E., "Cooperative Autonomous Distributed Robotic Exploration (CADRE)," *NASA*, 2021. URL http://www.nasa.gov/directorates/spacetech/game_changing_development/projects/CADRE.

[4] Staudinger, E., Shutin, D., Manß, C., Viseras, A., and Zhang, S., "Swarm Technologies For Future Space Exploration Missions," *14th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-sairas)*, 2018.

[5] Rouff, C., "Intelligence in Future NASA Swarm-based Missions," *AAIA Fall Symposium*, 2007.

[6] Buşoniu, L., Babuška, R., and De Schutter, B., "Multi-agent Reinforcement Learning: An Overview," *Innovations in Multi-Agent Systems and Applications - 1*, Vol. 310, edited by D. Srinivasan and L. C. Jain, Springer Berlin Heidelberg, 2010, pp. 183–221. https://doi.org/10.1007/978-3-642-14435-6_7, URL http://link.springer.com/10.1007/978-3-642-14435-6_7, series Title: Studies in Computational Intelligence.

[7] Oliehoek, F., and Amato, C., "A Concise Introduction to Decentralized POMDPs," 2016. https://doi.org/10.1007/978-3-319-28929-8.

[8] Bernstein, D. S., Zilberstein, S., and Immerman, N., "The Complexity of Decentralized Control of Markov Decision Processes," *Conference on Uncertainty in Artificial Intelligence*, 2000. URL https://api.semanticscholar.org/CorpusID:1195261.

[9] Huang, Y., Wu, S., Mu, Z., Long, X., Chu, S., and Zhao, G., "A Multi-agent Reinforcement Learning Method for Swarm Robots in Space Collaborative Exploration," 2020, pp. 139–144. https://doi.org/10.1109/ICCAR49639.2020.9107997.

[10] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y., "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games," *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[11] Zhang, K., Yang, Z., and Başar, T., "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," *ArXiv*, Vol. abs/1911.10635, 2019. URL https://api.semanticscholar.org/CorpusID:208268127.

[12] Dudukovich, R., Wagner, K., Kancharla, S., Fantl, J., and Fung, A., "Towards the Development of a Multi-Agent Cognitive Networking System for the Lunar Environment," *2021 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, IEEE, 2021, pp. 7–13. https://doi.org/10.1109/WiSEE50203.2021.9613839, URL https://ieeexplore.ieee.org/document/9613839/.

[13] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I., "Emergent Tool Use From Multi-Agent Autocurricula," *ArXiv*, Vol. abs/1909.07528, 2019. URL https://api.semanticscholar.org/CorpusID:202583612.

[14] Chen, Z., Subagdja, B., and Tan, A.-H., "End-to-end Deep Reinforcement Learning for Multi-agent Collaborative Exploration," *2019 IEEE International Conference on Agents (ICA)*, 2019, pp. 99–102. https://doi.org/10.1109/AGENTS.2019.8929192.

[15] Guan, Y., Zou, S., Peng, H., Ni, W., Yanglong, S., and Gao, H., "Cooperative UAV Trajectory Design for Disaster Area Emergency Communications: A Multiagent PPO Method," *IEEE Internet of Things Journal*, Vol. PP, 2023, pp. 1–1. https://doi.org/10.1109/JIOT.2023.3320796.

[16] Peña, J., Rouff, C., Hinchey, M., and Ruiz-Cortés, A., "Modeling NASA swarm-based systems: Using agent-oriented software engineering and formal methods," *Software and System Modeling*, Vol. 10, 2011, pp. 55–62. https://doi.org/10.1007/s10270-009-0135-2.

[17] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *CoRR*, Vol. abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

[18] Heess, N. M. O., Dhruva, T., Sriram, S., Lemmon, J., Merel, J., Wayne, G., Tassa, Y., Erez, T., Wang, Z., Eslami, S. M. A., Riedmiller, M. A., and Silver, D., "Emergence of Locomotion Behaviours in Rich Environments," *ArXiv*, Vol. abs/1707.02286, 2017. URL https://api.semanticscholar.org/CorpusID:30099687.