Project Final Report

Multi-Agent RL: Rethinking Observation and Policy Design under Cooperative MARL systems

Seouh-won Yi [1] **Sungwoo Cho** [1] **Daesoon Kim** [1]

1 Abstract

This paper investigates key challenges in *cooperative multi-agent reinforcement learning* (MARL): *observation complexity*, *partial observability*, and the trade-off between *decentralized* and *centralized* control. To address observation complexity, we compared MLP-based policies with graph-based architectures: *GNNs*, *EGNNs*, and *E2GN2*. Under partial observability, we validated the benefits of information sharing and incorporating inductive biases from agent-centric observations. While graph-based models did not consistently outperform MLPs, centralized policies exhibited superior stability and convergence, highlighting the importance of explicit feature inputs over purely learned representations. These findings underscore the need for thoughtful design in cooperative MARL systems.

2 Introduction

Reinforcement learning (RL) has traditionally focused on single-agent settings, where an individual agent learns to act optimally in a stationary environment based on its own interactions (Sutton & Barto, 2018). While successful in domains such as games, robotics, and control, single-agent RL is fundamentally limited in modeling scenarios that involve interaction among multiple decision-makers. To address the growing need for modeling social, collaborative, and competitive dynamics—common in real-world applications like autonomous driving, swarm robotics, and resource management—research has increasingly expanded into multi-agent reinforcement learning (MARL).

MARL involves multiple agents interacting within a shared environment, which may be cooperative, competitive, or mixed in nature (Lowe et al., 2017; Leibo et al., 2017). Depending on the task, agents may coordinate to achieve a common goal or compete for limited resources, and may or may not be able to explicitly communicate (Foerster et al., 2016; Sukhbaatar et al., 2016). These diverse settings give rise to unique challenges absent in single-agent RL, including partial observability, non-stationarity due to concurrently learning agents, and credit assignment in shared-reward scenarios. Effectively solving MARL tasks thus requires careful design of observation models and policy architectures that support scalable coordination and efficient information sharing under such constraints.

While recent work has proposed increasingly sophisticated architectures for MARL, including attention mechanisms (Iqbal & Sha, 2019; Zhang et al., 2021) and graph-based models (Jiang & Dun, 2019; Wang et al., 2021), relatively little attention has been paid to how the structure of observation space and control type (centralized vs. decentralized) jointly affect performance under partial observability. In practice, observation design and representation learning are often entangled with policy learning, making it difficult to assess their individual contributions to coordination and sample efficiency.

In this work, we isolate and compare the effects of observation modeling and control strategy on cooperative MARL. Specifically, we examine (1) how different observation encodings—absolute vs. relative coordinates, agent-centric vs. global views—affect performance, and (2) how graph-based policy architectures, such as GNNs and EGNNs, improve coordination in partially observable settings. We implement and benchmark these variants under both centralized and decentralized training protocols.

By conducting systematic evaluations across two distinct environments, we aim to provide actionable in-

[1]Department of Data Science, Seoul National University.

sights into when and how structured representations benefit cooperative MARL. Our findings suggest that careful observation modeling and equivariant architectures can significantly enhance learning stability and coordination, especially in decentralized settings where information is sparse or noisy. These results are not only practically relevant for real-world multi-agent deployments, but also offer new perspectives on modular policy design for scalable MARL.

The report is organized as follows: Section 3 reviews related work, Section 4 provides background on MARL and MA-POMDPs, Section 5 describes our experiments, Section 6 presents experimental results, and Section 7 concludes with future directions.

3 Related Works

Multi-agent reinforcement learning (MARL) has attracted considerable interest due to its applicability in domains such as autonomous driving, swarm robotics, distributed sensor networks, and multi-agent games. Early research in MARL largely assumed fully observable settings, which simplified coordination and learning. However, many real-world applications involve partial observability, where agents must make decisions based on limited local information. These challenges are formalized within the Multi-Agent Partially Observable Markov Decision Process (MA-POMDP) framework (Oliehoek & Amato, 2016), which captures the inherent difficulties of decentralized control, uncertainty, and communication bottlenecks.

To address the issue of coordination under partial observability, the Centralized Training with Decentralized Execution (CTDE) paradigm has emerged as a widely used training protocol (Lowe et al., 2017; Foerster et al., 2018). In this framework, agents are trained using centralized critics with access to global state information, but they must act based solely on local observations during deployment. This setup has led to the development of influential algorithms such as MADDPG(Lowe et al., 2017), QMIX(Rashid et al., 2018), and COMA (Foerster et al., 2018), which demonstrate improved coordination in cooperative settings. Nonetheless, CTDE methods still struggle with generalization and scalability, especially when moving to fully decentralized settings where agents learn independently without centralized critics.

To enhance information sharing and structural reasoning among agents, recent works have incorporated Graph Neural Networks (GNNs) into MARL. GNNs exploit the relational inductive biases between agents by modeling their interactions as a graph, where nodes represent agents and edges capture dynamic or spatial relationships (Battaglia et al., 2018; Jiang & Dun, 2019). By passing messages through this graph structure, agents can learn to coordinate in a more structured and sample-efficient manner. Moreover, Equivariant Graph Neural Networks (EGNNs) extend this idea by encoding symmetry and permutation invariance in multi-agent observations, improving generalization and reducing sample complexity (Wang et al., 2021; Satorras et al., 2021).

Building on this, recent models such as E2GN2 (McClellan et al., 2024) leverage equivariant architectures and targeted exploration strategies to further enhance learning in sparse-reward or partially observable environments. However, while graph-based models have shown promise, their relative advantages over simpler architectures like MLPs in fully cooperative settings remain underexplored. In particular, the interplay between observation modeling (e.g., absolute vs. relative encoding) and architecture choice is not well understood. Understanding this interplay is crucial for designing scalable MARL systems that can generalize across tasks with varying observability requirements.

4 Preliminaries

4.1 Multi-Agent POMDPs

Cooperative multi-agent reinforcement learning (MARL) problems are frequently modeled using the Multi-Agent Partially Observable Markov Decision Process (MA-POMDP) framework (Oliehoek & Amato, 2016). A MA-POMDP is defined by the tuple:

$$(\mathcal{I}, \mathcal{S}, \{\mathcal{A}_i\}, \mathcal{P}, R, \{\mathcal{O}_i\}, \mathcal{O}, \gamma)\,,$$

where $\mathcal{I}$ denotes the set of N agents, $\mathcal{S}$ is the global state space, $\mathcal{A}_i$ and $\mathcal{O}_i$ are the action and observation spaces of agent i, $\mathcal{P}$ is the transition function, R is the shared reward function, $\mathcal{O}$ is the observation function, and $\gamma \in [0, 1)$ is the discount factor.

At each timestep t, each agent i receives a private observation $o_i^t \sim \mathcal{O}_i(\cdot \mid s_t)$, which provides only partial information about the underlying global state s_t. This

partial observability reflects practical constraints in real-world environments, where agents may have limited sensing capabilities or communication bandwidth. Consequently, agents must make decisions under uncertainty and rely on incomplete, local views of the environment.

Each agent selects an action $a_i^t \sim \pi_i(a_i^t \mid h_i^t)$ based on its local action-observation history h_i^t. The environment then transitions to a new state $s_{t+1} \sim \mathcal{P}(\cdot \mid s_t, \mathbf{a}_t)$, where $\mathbf{a}_t = (a_1^t, \ldots, a_N^t)$ denotes the joint action. All agents receive a shared reward $r_t = R(s_t, \mathbf{a}_t)$.

The objective is to find a joint policy $\boldsymbol{\pi} = \{\pi_i\}_{i \in \mathcal{I}}$ that maximizes the expected cumulative return:

$$\mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=1}^{T} \gamma^{t-1} R(s_t, \mathbf{a}_t) \right].$$

Under partial observability and decentralized decision-making, solving MA-POMDPs is significantly more challenging than single-agent MDPs, requiring both robust local policies and mechanisms for coordination.

4.2 Centralized vs. Decentralized Learning

A key design axis in MARL is the degree of centralization during training and execution. A widely adopted approach is *Centralized Training with Decentralized Execution* (CTDE) (Lowe et al., 2017), in which agents are trained with access to additional global state information or centralized critics, but execute using only local observations. This paradigm balances the expressiveness of centralized models with the scalability and flexibility of decentralized execution.

By contrast, fully decentralized methods constrain both training and execution to local observations. While more realistic for real-world systems with limited communication or privacy constraints, such methods face greater challenges in coordination and stability (Foerster et al., 2016; Iqbal & Sha, 2019).

Hybrid methods also exist, where explicit inter-agent communication is learned (Sukhbaatar et al., 2016; Das et al., 2019), or implicit coordination emerges from shared inductive biases or shared policy parameters.

4.3 Key Challenges in Cooperative MARL

Cooperative MARL settings introduce a number of challenges not present in single-agent RL:

- **Decentralized Decision-Making**: Agents must act based solely on local information, limiting coordinated behavior.
- **Partial Observability**: Limited observations hinder globally consistent decisions.
- **Non-Stationarity**: Other agents' evolving policies make the environment unstable.
- **Credit Assignment**: Identifying individual contributions to a shared reward remains difficult without structured modeling.
- **High-Dimensional Observations**:Multi-agent settings often limits the effectiveness of standard algorithms on single-agent environments.

These challenges have motivated the design of relational policy architectures (e.g., GNNs (Battaglia et al., 2018)), attention-based mechanisms (Iqbal & Sha, 2019), and symmetry-aware models (e.g., EGNNs (Wang et al., 2021)) that facilitate generalization.

5 Experiments

Our method addresses three core research questions related to cooperative multi-agent reinforcement learning (MARL): improving sample efficiency through graph-based models, evaluating policy learning in partially observable environments, and comparing centralized vs. decentralized control.

5.1 Improvement via Graph-based Encoders

To address the issue of sample complexity in multi-agent environments, we implemented and compared four actor policy architectures. The first is a standard MLP-based policy trained with Proximal Policy Optimization (PPO) (Schulman et al., 2017). The second employs a Graph Neural Network (GNN) that encodes agent interactions as a graph. The third is an Equivariant GNN (EGNN) that incorporates symmetry-aware representations (Wang et al., 2021). Lastly, we evaluate E2GN2, an exploration-enhanced variant of EGNN that aims to improve sample efficiency through structured exploration (McClellan et al., 2024).

These models introduce relational and symmetry-based inductive biases that help reduce redundant state exploration. We evaluated their performance in both

decentralized and centralized settings to assess their impact on learning efficiency.

5.2 `Area Coverage`: MA-POMDP Benchmark

Existing MARL benchmarks often lack environments that explicitly capture partial observability and realistic agent limitations. To address this, we introduce a grid-based MA-POMDP environment called *Area Coverage*, inspired by real-world cleaning robots. Each agent in the environment represents a robot with a unidirectional suction mechanism and is tasked with cooperatively covering all non-wall grid cells.

Agents can choose between three discrete actions: `Stop`, `Rotate`, and `Move Forward`. At each step, an agent receives a partial observation consisting of a 3-cell-wide forward-facing cone centered on its current location. The observation space thus reflects realistic sensing constraints in spatially-structured domains.

We evaluate the environment under two settings: one with a fixed map layout and deterministic initial agent positions, and another with randomized tiles and agent initialization per episode. This dual configuration enables the study of policy robustness and generalization under varying levels of environmental complexity.

Reward and Observation Design The reward structure combines global cooperation and local incentives. All agents receive a +100 global reward upon full coverage, and a time penalty of -0.05 per step. Additionally, local rewards are provided to promote efficiency and avoid conflicts: -1 for wall collisions, -2 for collisions with other agents, $+5$ for newly covered tiles, and -0.1 for revisiting already covered cells. Each agent's observation is mapped to a tensor of shape $(\text{grid}_y, \text{grid}_x, 5)$, encoding tile-level features as well as absolute and relative positions.

Observation Encoding with Priors To facilitate generalization, we preprocess raw observations using spatial priors. Specifically, the observation is padded and rotated such that the agent's facing direction is aligned with a canonical (default) orientation regardless of its actual pose. The centered and aligned observation is then cropped, and each grid cell is mapped through a learnable embedding layer that encodes tile type and positional information. This structured representation is then passed to the policy network.

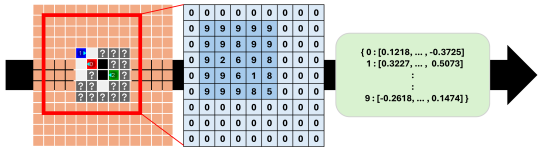


Figure 1: Map-aware observation embedding structure that incorporates orientation-based spatial priors.

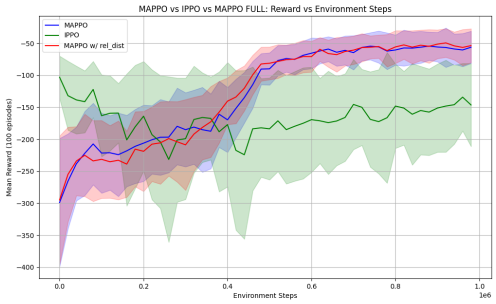
In the fixed map configuration of the *Area Coverage* environment, policies augmented with spatial priors exhibited significantly faster convergence and more coherent coordination than those using unprocessed inputs. These priors helped agents generalize across symmetric situations and reduced redundant behaviors during coverage. In the randomized map setting, where tile layouts and initial agent positions vary across episodes, the benefits of such priors were less immediate but still notable. Although convergence was slower and more variable, policies leveraging structured encodings maintained better spatial coverage and reduced overlap, indicating more stable and consistent learning behavior under uncertainty.

As detailed in Appendix A, these results suggest that incorporating environment-specific priors into the observation encoding process can enhance both robustness and generalization of cooperative policies across varying environmental complexities.

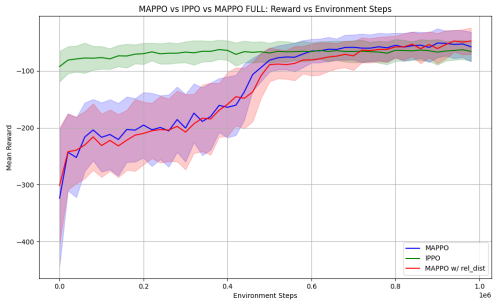
5.3 Centralized vs. Decentralized Control

We evaluated centralized versus decentralized policy learning in the MPE *Simple Spread* environment. Centralized agents were trained using full global state information, while decentralized agents relied solely on local observations without access to other agents' states or actions.

All experiments were run for 1 million timesteps, using consistent training schedules across settings. To investigate the effect of observation encoding, we compared two variants: absolute coordinates and relative coordinates (e.g., distances to landmarks and other agents). This allowed us to isolate the role of spatial encoding under both control paradigms. Full details on network architectures, optimization parameters, and observation preprocessing are provided in Appendix A.



(a) Absolute observations



(b) Relative observations

Figure 2: Comparison of decentralized PPO performance in MPE under different observation schemes.

6 Results

In this section, we present empirical results addressing the three research questions, structured in alignment with our experimental setup.

6.1 Graph-based vs. MLP Architectures

We evaluated whether graph-based policy architectures offer improved sample efficiency compared to the MLP baseline. Across both centralized and decentralized settings, all models—MLP, GNN, EGNN, and E2GN2—exhibited similar trends in convergence speed and final coverage performance (see Figure 3). Although EGNN and E2GN2 incorporate symmetry-aware inductive biases, they did not consistently outperform the MLP baseline. This suggests that for tasks of moderate complexity, simply introducing graph structure may not yield significant benefits, and additional equivariant mechanisms may be required to fully exploit the relational inductive bias.

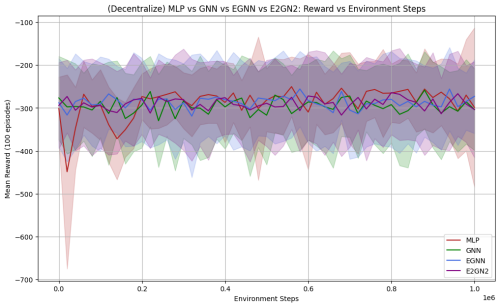


Figure 3: Cumulative rewards on Simple Spread.

6.2 Comparing Control Paradigms

In the MPE Simple Spread environment, centralized policies significantly outperformed decentralized policies across all observation schemes. Centralized PPO agents reliably learned to coordinate and achieve landmark coverage with lower variance. Among decentralized agents, those trained with relative coordinates (e.g., distances to landmarks) showed better stability and reward growth than those with only absolute observations, though the performance gap to centralized policies remained.

7 Conclusion

We studied how observation structure and policy design affect performance in cooperative multi-agent reinforcement learning. Through controlled experiments in partially observable environments, we compared centralized and decentralized policies using MLPs, GNNs, EGNNs, and E2GN2. Our results show that centralized policies consistently outperform decentralized ones in terms of stability and sample efficiency. Relative observation encoding notably improved performance in decentralized settings. Surprisingly, graph-based encoders including EGNN and E2GN2 did not outperform simple MLPs, indicating that the benefits of equivariance and relational bias by only previous approaches are task-dependent and may not generalize to all cooperative settings. These findings emphasize the importance of observation modeling and control structure over model complexity. Future work should investigate how to better align architectural inductive biases with environment-specific demands.

References

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., and Pineau, J. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, 2019.

Foerster, J., Assael, Y. M., de Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Iqbal, S. and Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 2961–2970, 2019.

Jiang, J. and Dun, Z. Graph convolutional reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. Multi-agent reinforcement learning in sequential social dilemmas. *Proceedings of the 16th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2017.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 2017.

McClellan, A., Zhang, L., and Kumar, V. Boosting exploration in multi-agent reinforcement learning with equivariant graph neural networks. *arXiv preprint arXiv:2401.09876*, 2024.

Oliehoek, F. A. and Amato, C. A concise introduction to decentralized pomdps. *SpringerBriefs in Intelligent Systems*, 2016.

Rashid, T., Samvelyan, M., Schroeder de Witt, C., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *International Conference on Machine Learning*, 2018.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. *ICML*, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sukhbaatar, S., Fergus, R., et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT press, 2018.

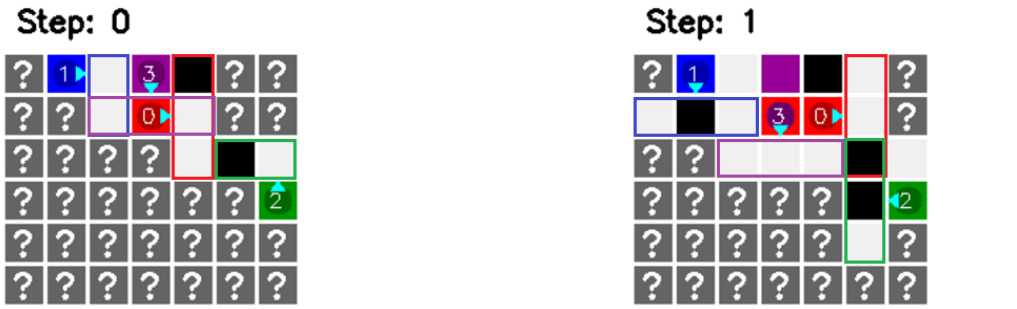
Wang, R., Chen, X., and Li, Z. Equivariant graph neural networks for multi-agent reinforcement learning. *arXiv preprint arXiv:2106.12345*, 2021.

Zhang, M., Wang, Y., Zhou, H., Li, Z., and Yang, Q. Multi-agent attention critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 20470–20481, 2021.

A Cooperative Cleaning Environment Design

A.1 Environment Specification of the `Area Coverage` Benchmark

We introduce a custom multi-agent environment called *Cooperative Cleaning*, in which a team of cleaning robots must collaboratively cover all reachable (non-wall) tiles on a grid. The objective is to minimize the total number of steps taken, incentivizing efficient and coordinated exploration.



(a) Initial map layout with agents and obstacles.

(b) Agent positions and covered tiles after 1 step.

Figure 4: Example of the Cooperative Cleaning environment.

Environment Specification

- **Agents:** Each agent is assigned a unique ID and acts synchronously on a discrete 2D grid.
- **Action Space (4 discrete actions):** `Stop`, `Move Forward`, `Turn 90° Left`, and `Turn 90° Right`.
- **Obstacles and Collisions:** Black cells represent impassable walls. An agent attempting to move into a wall remains in place. If two agents move into the same cell simultaneously, both revert to their previous positions.
- **Observations:** Each agent receives a partial observation consisting of a 3-tile cone in front of it. The observation is encoded as a tensor over the grid, with each tile assigned one of the following states:

  0: Outside the grid
  1: Agent itself
  2–5: Other agents, with relative orientation (same, right, left, back)
  6: Wall
  7: Visited and covered tile
  8: Seen but not covered tile
  9: Unknown

- **Reward Structure:** Each agent receives a local reward at every timestep, and all agents share a global reward when the task is completed.
  - *Global Reward:*

$$r_{\text{global}} = \begin{cases} +100, & \text{if all cleanable tiles } \mathcal{D} \text{ are covered} \\ 0, & \text{otherwise} \end{cases} \quad - 0.05 \text{ (time penalty per step)}$$

– *Local Reward (per agent):*

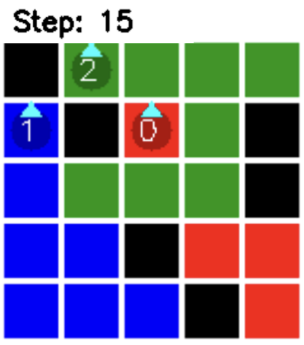
$$r = \begin{cases} +5, & \text{new tile cleaned} \\ -0.01, & \text{already cleaned tile revisited} \\ -2, & \text{collision with another agent} \\ -1, & \text{collision with wall} \end{cases}$$

Agents receive the sum of all applicable rewards at each step.

- **Termination:** The episode ends when all tiles are cleaned or a maximum number of steps is reached.
- **Generalization:** The environment supports variable grid sizes, number of agents, obstacle densities, and randomized initializations, allowing for robust generalization testing.

A.2 Numerical Results in the `Area Coverage` Environment

To validate the effectiveness of our environment design and evaluate whether standard PPO agents can learn effective strategies under partial observability, we conducted experiments using our `Area Coverage` benchmark. All experiments were run with $N = 3$ agents on a 5×5 grid. Agents were initialized with random positions and orientations. Walls were randomly placed to cover 20% of the grid, while ensuring the environment remained fully navigable. We trained agents using a PPO algorithm with a 2-layer MLP policy (each layer with 64 hidden units for each Actor and Critic networks), for 1,000 epochs. The rollout length was set to 500, and training was performed using 10 parallel environments per batch.



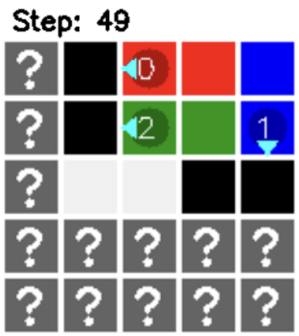
(a) Trajectory of PPO-trained agents in the fixed 5×5 map.

(b) Trajectory of PPO-trained agents in the fixed 12×12 map.

Figure 5: Comparison of PPO agent behavior in randomized environments. (a) The policy successfully learned the optimal behavior, enabling the agents to cover the entire map in 15 steps. (b) The agents learned to divide the map and explore their designated regions (green: bottom-right, blue: bottom-left, red: top). Afterwards, they return and efficiently fill in the remaining cells that other agents failed to cover.

A.2.1 Fixed Environment Setting

In the fixed-map setting, the environment was initialized once using a fixed random seed, and held constant throughout training. As shown in Figure 5, PPO-trained agents successfully learned coordinated coverage strategies. Compared to a random policy baseline, the trained policy achieved full coverage in approximately 15 steps while avoiding wall and agent collisions. Agents learned to split directions effectively, minimizing redundancy and overlap in their trajectories.



(a) Policy trained on raw observations.

(b) Policy trained with preprocessed observation priors.

Figure 6: Comparison of PPO agent behavior in randomized environments. When using the raw observation as is (a), the policy failed to generalize well even after 1000 epochs of training. In contrast, when the observation was processed into a more agent-friendly format (b), the policy achieved a certain level of generalization — albeit imperfect — after just 100 epochs.

We also scaled the grid to 12×12 to test the learned policy's adaptability. Despite the increased complexity, the agents converged to a coordinated strategy where each agent covered a distinct region of the map before revisiting any uncovered areas left behind by others, demonstrating learned division of labor and efficient navigation.

A.2.2 Arbitrary Dynamic Environments

In contrast to the fixed setting, we also tested policies in a dynamic scenario where maps, wall locations, and agent initial positions/orientations were randomized at each episode. This setting was designed to evaluate policy generalization under varying layouts.

As expected, PPO policies trained directly on raw observations struggled to generalize, failing to converge to effective behaviors across variable maps (Figure 6a). We hypothesize this is due to high variance in the state-action distribution and insufficient trajectory overlap across environments, which destabilizes replay-based learning.

To mitigate this, we applied observation preprocessing using rotation-aware padding and cropping, followed by token embedding of grid states. This encoded orientation-invariant local observations into a normalized form. With this representation, agents exhibited noticeable improvements in learning, achieving moderately successful coverage behavior after only 100 epochs, even in previously unseen maps (Figure 6b).

These results underscore the importance of task-specific observation design in partially observable settings. Encoding structured priors helped generalize across layouts, suggesting that further improvements could be achieved by integrating graph-based modeling or convolutional encoders tailored for spatial inductive biases.

B Implementation and Hyperparameter Details

Hyperparameters	Value
Train batch size	2000
Mini-batch size	1000
PPO clip	0.2
Learning rate	3×10^{-5}
Number of SGD iterations	10
γ (discount factor)	0.99
λ (GAE parameter)	0.95

Table 1: Hyperparameters used for PPO training in the MPE Simple Spread environment.

Implementation Details Our environment and agents are implemented in Python, leveraging the PyTorch deep-learning framework alongside the PettingZoo and MPE libraries for standardized multi-agent interfaces. All experiments are conducted on 2 NVIDIA RTX 3090 GPU, ensuring fast training and simulation. To facilitate qualitative analysis, we built a Matplotlib-based renderer that produces trajectory visualizations of agent movements and interactions.