

Article

Decentralized Shared Actor–Critic Learning for Collision-Aware Small-Team Multi-Robot Coverage

Abzal E. Kyzyrkanov ^{1,*}, Didar Yedilkhan ^{1,*}, Saltanat Amirgaliyeva ^{2,*} and Sergazy Narynov ²

¹ Smart City Research Center, Astana IT University, Astana 010000, Kazakhstan

² Institute of Information and Computational Technologies, Almaty 050010, Kazakhstan

* Correspondence: d.yedilkhan@astanait.edu.kz (D.Y.); s.amirgaliyeva@gmail.com (S.A.)

Abstract

This study presents a decentralized shared actor–critic framework for cooperative multi-robot coverage in continuous two-dimensional simulation. The method combines permutation-invariant local observations, continuous differential-drive control, and reward shaping based on stepwise Hungarian assignment distances, collision penalties, and time efficiency. Homogeneous teams of four, five, and six agents are evaluated in an obstacle-free environment using five independent training seeds. In the final training window, the full reward configuration achieved full-team success rates of $98.2 \pm 2.9\%$ for four agents, $85.1 \pm 18.0\%$ for five agents, and $96.3 \pm 2.0\%$ for six agents, with mean landmark coverage above 96% in all cases. The lower mean in the five-agent setting was associated with higher seed-level variability dominated by one low-success seed. Reward ablations without assignment shaping or collision penalties remained viable, and seed-level tests did not show a statistically significant final-window advantage of the full reward configuration. The full configuration reached the 80% rolling-success threshold earlier in median terms, with the clearest seed-level support in the four-agent setting. Within-environment comparison showed higher full-team success than MADDPG and MAPPO under the matched training horizon and final-window protocol. Deterministic arena-size transfer from 15×15 to 30×30 showed decreasing full-team success as arena size increased, while partial landmark coverage remained higher than strict full-team completion. The results support the method for small homogeneous teams in the tested obstacle-free simulation, while larger teams, external obstacles, aerial-robot dynamics, formal safety guarantees, and hardware deployment remain future work.

Keywords: multi-robot systems; multi-agent reinforcement learning; shared actor–critic; decentralized coordination; task allocation; collision-aware motion; continuous control; cooperative coverage



Academic Editor: Beno Benhabib

Received: 22 April 2026

Revised: 14 June 2026

Accepted: 23 June 2026

Published: 25 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Reinforcement learning (RL) and deep reinforcement learning (DRL) provide a framework for sequential robotic decision making in uncertain and continuous environments [1–6]. In multi-agent reinforcement learning (MARL), several agents learn and act simultaneously, which introduces non-stationarity, partial observability, coordination difficulty, and credit-assignment challenges [7–12]. These challenges are directly relevant to cooperative multi-robot coverage, where robots must distribute themselves over spatially separated targets while limiting interference and inefficient motion.

The problem is relevant across ground and aerial multi-robot systems. Multi-agent strategies are increasingly studied for connected and automated vehicles [13], decentralized multi-robot coordination [14], and swarm robotics [15,16]. Related application-oriented work includes intelligent perception pipelines [17], self-organizing mobile-robot frameworks [18], and behavior-based or fuzzy-logic coordination strategies for swarm motion and terrain monitoring [19–24]. Aerial multi-agent systems further motivate coverage research, including drone monitoring, decentralized UAV control, obstacle-aware navigation, route optimization, and multi-UAV planning taxonomies [25–30]. However, UAV-specific dynamics, sensing, communication, safety constraints, and hardware validation are outside the scope of this study. The present work evaluates an abstract obstacle-free two-dimensional simulation rather than a platform-specific deployment.

Several MARL algorithm families define the methodological context. MADDPG uses centralized critic information during training while retaining decentralized actors at execution time [31]; VDN and QMIX decompose or factorize team value functions [32,33]; COMA addresses cooperative credit assignment [34]; and PPO-based methods such as MAPPO and IPPO have shown strong empirical performance in cooperative benchmarks [35–37]. In continuous-control actor–critic learning, DDPG provides a deterministic off-policy foundation, while TD3 and SAC address related stability and exploration issues [3,38,39]. Standard cooperative MARL environments such as SMAC, SMACv2, and VMAS provide broader benchmark contexts [40–42]. This paper instead studies a custom continuous-control coverage task and includes direct within-environment comparisons with MADDPG and MAPPO; broader benchmark evaluation is left for future work.

The research gap lies in coupling assignment, coverage, and collision-aware continuous motion within a decentralized learned policy. Multi-robot task allocation (MRTA) remains difficult because robots must be assigned to goals or subtasks under resource, time, and coordination constraints [43–46], with additional computational structure studied through complexity and greedy-allocation analyses [47,48]. Collision-aware behavior remains difficult under dense interactions, especially in continuous space [49–52]. Coverage also becomes harder as team size grows because completeness, efficiency, collision avoidance, and interference must be balanced simultaneously [53]. Multi-agent path finding (MAPF) offers well-defined graph-based formulations and benchmarks [54–61], including learning-based variants such as PRIMAL and SCRIMP [62,63]; however, graph-based MAPF remains distinct from continuous, dynamics-aware robotic coverage. Recent work on coverage-oriented DRL and curriculum-based MARL further indicates that stable multi-robot learning often requires structured training rather than naive end-to-end optimization [64–67]. Stability and robustness also remain important in broader RL-controlled multi-component systems [68].

Existing approaches therefore often emphasize one aspect of the problem at the expense of another: graph planners provide combinatorial structure but abstract away continuous execution; centralized learners can optimize efficiently but may introduce scaling and robustness concerns; and behavior-based or fuzzy-rule systems can be efficient but often remain task-specific or manually designed [19–21,23,29]. In contrast, this study targets a local-observation-based formulation in which assignment and collision-aware motion are coupled within a single learned policy operating directly in continuous state–action spaces. The novelty is not any individual component alone, but the specific integration of assignment-aware reward shaping, shared local actor–critic learning, permutation-invariant observations, and continuous differential-drive coverage evaluation.

Accordingly, the aim of this research is to develop and evaluate a decentralized learning framework for cooperative coverage in which homogeneous agents share a common actor–critic structure, observe the environment through local permutation-invariant repre-

sentations, and receive assignment-aware guidance through stepwise Hungarian matching. The work addresses three questions: whether dynamic goal assignment and collision-aware motion can be integrated into one continuous-control policy, whether the formulation remains effective over the evaluated team sizes $n \in \{4, 5, 6\}$, and whether decentralized execution can be preserved in the tested simulated setting. These questions place this study at the intersection of MARL, MRTA, small-team multi-robot coordination, and continuous robotic navigation [36,41,69–78].

The experiments evaluate this formulation within a deliberately limited scope: homogeneous differential-drive agents, team sizes $n = 4, 5, 6$, and an obstacle-free continuous two-dimensional coverage environment. The evaluation includes multi-seed training, reward ablations, HER/restart training-pipeline ablations, within-environment MADDPG and MAPPO baselines, deterministic arena-size transfer, sample-efficiency diagnostics, direct collision metrics, path-length diagnostics, and computational-cost reporting. The contribution is therefore an empirical analysis of an integrated shared actor–critic framework for small-team simulated multi-robot coverage; larger teams, obstacle-rich environments, platform-specific UAV dynamics, standard MARL benchmark suites, and physical deployment remain future work.

2. Materials and Methods

2.1. Problem Formulation

2.1.1. Task Definition

The task under consideration involves n cooperative agents operating in a continuous two-dimensional environment containing n fixed target locations, hereafter referred to as landmarks. At the beginning of each episode, both agents and landmarks are positioned randomly within the environment without overlap. The objective is for all agents to reach and remain at assigned landmarks while minimizing the total distance traveled by the group.

The coordination problem is addressed in a simplified simulation setting. Each agent is represented by its planar position for assignment and observation purposes; collision checking uses a finite agent radius, and motion follows the differential-drive kinematic model described in Section 2.2.3. This abstraction allows this study to focus on the coordination and assignment mechanisms in an obstacle-free simulated coverage task. Extensions to platform-specific dynamics, richer sensing, external obstacles, and physical robots are outside the present evaluation.

2.1.2. Environment Design

The environment is modeled as a continuous two-dimensional plane with an environment-size parameter L used for initialization and coordinate normalization. Agents and landmarks are initialized inside the $L \times L$ sampling domain, but agent positions are not clipped to this domain during an episode. Each agent is represented as a circular entity with a fixed radius and is capable of executing continuous velocity commands in the plane. Landmarks are modeled as points, each representing a target location that must be covered by exactly one agent.

No external static obstacles are included in the experiments reported in this study. Collision checking is therefore performed among agents and, after completion, inactive agents may remain as occupied entities in the environment. Obstacle-rich environments are outside the scope of the present evaluation.

At the start of each episode, the positions of agents and landmarks are initialized randomly within the $L \times L$ sampling domain, ensuring that no initial overlaps occur. The episode proceeds in discrete time steps, during which agents move according to

their control inputs and may move outside the initial sampling domain. The simulation terminates when either all landmarks have been reached by their assigned agents or a predefined maximum number of steps is exceeded.

2.2. Reinforcement Learning

Reinforcement learning (RL) is a computational framework for sequential decision-making, where autonomous agents learn to act in an environment through trial-and-error interaction [2]. At each discrete time step, an agent perceives the environment through an observation, selects an action according to its policy, and receives both a scalar reward and the subsequent observation as feedback from the environment. The policy is updated so as to maximize the expected long-term return, usually defined as the discounted cumulative sum of rewards. This closed feedback loop between perception, action, and reward enables the emergence of adaptive behavior without explicit supervision.

In multi-agent reinforcement learning (MARL), several agents operate concurrently in a shared environment, each with its own stream of observations, actions, and rewards. Figure 1 illustrates this interaction, where each agent transmits an action to the environment and in return receives an observation–reward pair. The setting naturally extends the single-agent case but introduces additional challenges related to coordination, partial observability, and non-stationarity induced by the simultaneous adaptation of multiple policies [79]. Addressing these challenges requires careful problem formulation and learning architectures that support stable training and evaluation across team sizes.

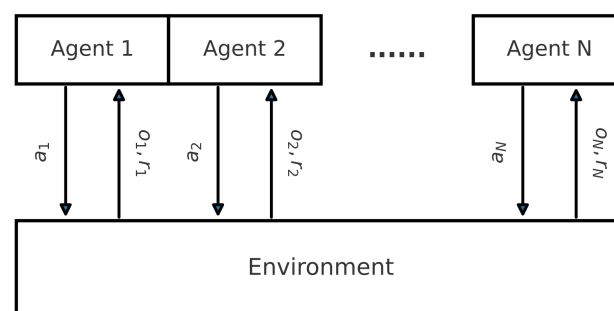


Figure 1. Multi-agent interaction with the environment. Each agent $i \in \{1, \dots, N\}$ selects an action a_i and receives an observation–reward pair (o_i, r_i) from the environment.

2.2.1. Markov Decision Process

A Markov decision process (MDP) provides a formal foundation for reinforcement learning by modeling sequential decision making under uncertainty. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of possible actions, $P(s' | s, a)$ denotes the probability of transitioning from state s to state s' after taking action a , $R(s, a)$ represents the expected reward for that transition, and $\gamma \in [0, 1)$ is the discount factor that prioritizes immediate rewards over distant ones [1]. A policy $\pi(a | s)$ specifies the behavior of an agent in terms of action selection given current state. The goal is to determine a policy that maximizes the expected cumulative discounted reward:

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right].$$

In the multi-agent domain, each agent i interacts with the shared environment independently, with its own local state s_i , action a_i , and reward r_i . A critic estimates the action-value function $Q_i(s_i, a_i)$, defined by the Bellman equation:

$$Q_i(s_i, a_i) = \mathbb{E} [r_i + \gamma Q_i(s'_i, a'_i) | s_i, a_i],$$

where s'_i and a'_i are the subsequent state and action of agent i . A shared actor delivers a common policy $\pi(a_i | s_i)$ across all agents. A shared actor provides a common policy parameterization across homogeneous agents, while the local critic avoids the complexity of joint-action value estimation. Coordinated behavior arises implicitly through shared parameter learning rather than explicit centralized action evaluation.

2.2.2. State Space

The state space of each agent coincides with its observation space, since every agent directly perceives the relevant information from the environment without additional hidden variables. Figure 2 illustrates the geometry of the observation process for an observing agent (red), another agent (blue), and a landmark (green), together with their projections in both the global coordinate system (X, Y) and the observer's local coordinate system (X', Y') . Each agent i occupies a position (X_i, Y_i) in the global frame and is oriented with heading θ_i relative to the X -axis. The observing agent establishes its local coordinate system by aligning the X' -axis with its own heading direction θ_i , while the Y' -axis is defined orthogonally. A point (X, Y) expressed in the global frame is transformed into the observer's local frame according to

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} \cos \theta_i & \sin \theta_i \\ -\sin \theta_i & \cos \theta_i \end{bmatrix} \begin{bmatrix} X - X_i \\ Y - Y_i \end{bmatrix}.$$

Consequently, the position of the landmark (X_l, Y_l) is represented in local coordinates as (X'_l, Y'_l) , and the position of the other agent (X_a, Y_a) as (X'_a, Y'_a) . The orientation of another agent a , specified globally as θ_a , is expressed in the local frame of the observer by the relative heading

$$\theta' = \theta_a - \theta_i,$$

which is subsequently normalized through its trigonometric representation

$$(\cos \theta', \sin \theta'),$$

in order to avoid discontinuities at the boundaries $\pm\pi$ and to provide a continuous encoding. To achieve scale invariance with respect to the physical dimensions of the environment, all relative coordinates are normalized by the environment size L such that $X' \mapsto X'/L$ and $Y' \mapsto Y'/L$. This normalization is applied consistently to landmarks and to other agents. The state of the observing agent therefore comprises its own heading encoded as $(\cos \theta_i, \sin \theta_i)$, the normalized relative positions of all landmarks and agents in its local frame, and the relative headings of all other agents encoded as $(\cos \theta', \sin \theta')$. Since both agents and landmarks are interchangeable entities, their order within the state vector would otherwise be arbitrary. To prevent ambiguity and to ensure invariance to permutations across different episodes, landmarks and agents are sorted according to their relative distance to the observing agent before encoding them into the state vector. The agents evolve according to differential-drive kinematics, where each agent executes an action $a = (v, \omega)$ consisting of a linear velocity v and an angular velocity ω . The resulting dynamics are described by

$$X_i(t+1) = X_i(t) + v \cos(\theta_i), \quad Y_i(t+1) = Y_i(t) + v \sin(\theta_i), \quad \theta_i(t+1) = \theta_i(t) + \omega.$$

As shown in Figure 2, the observing agent transforms the global positions of the landmark and the other agent into its egocentric frame, computes the relative heading of the other

agent, and normalizes all quantities to obtain a compact, continuous, and permutation-invariant state representation.

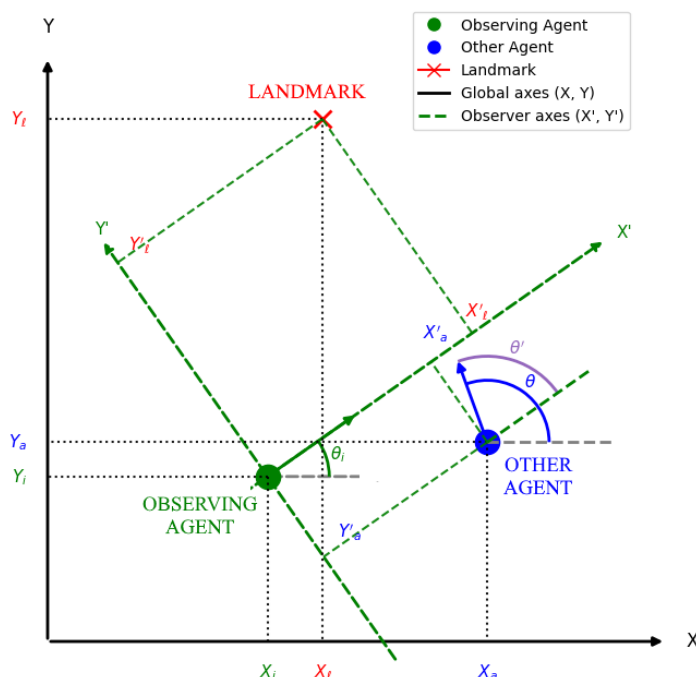


Figure 2. Geometric construction of the observation process. The observing agent (red) establishes its local coordinate system (X', Y') aligned with its heading θ_i . The positions of the landmark (green) and the other agent (blue) are projected into this frame, yielding (X'_l, Y'_l) and (X'_a, Y'_a) , respectively. The relative orientation of the other agent is given by $\theta' = \theta_a - \theta_i$.

2.2.3. Action Space

The control of each differential-drive agent is parameterized directly by its linear and angular velocities. For agent i , the action at discrete time t is

$$a_i(t) = (v_i(t), \omega_i(t)),$$

where $v_i(t)$ is the forward (tangential) linear velocity, and $\omega_i(t)$ is the yaw rate. In the implemented environment, the policy outputs a normalized action $\hat{a}_i(t) = (\hat{v}_i(t), \hat{\omega}_i(t)) \in [-1, 1]^2$ which is mapped to the physical controls by

$$v_i(t) = \frac{1}{2}(\hat{v}_i(t) + 1) v_{\max}, \quad \omega_i(t) = \hat{\omega}_i(t) \omega_{\max}.$$

Consequently,

$$v_i(t) \in [0, v_{\max}], \quad \omega_i(t) \in [-\omega_{\max}, \omega_{\max}],$$

with the constants fixed to $v_{\max} = 1$ and $\omega_{\max} = \frac{\pi}{2}$ in the reported experiments.

Let $(x_i(t), y_i(t)) \in \mathbb{R}^2$ denote the global position of agent i , and let $\theta_i(t) \in (-\pi, \pi]$ be its heading in the global frame. With the above action parameterization, the discrete-time kinematics used by the environment update positions and heading as

$$\begin{aligned} x_i(t+1) &= x_i(t) + v_i(t) \cos(\theta_i(t)), & y_i(t+1) &= y_i(t) + v_i(t) \sin(\theta_i(t)), \\ \theta_i(t+1) &= \text{wrap}_{(-\pi, \pi]}(\theta_i(t) + \omega_i(t)), \end{aligned}$$

where $\text{wrap}_{(-\pi, \pi]}(\cdot)$ denotes angle wrapping into $(-\pi, \pi]$; in implementation this is realized by modular arithmetic:

$$\text{wrap}_{(-\pi, \pi]}(\phi) = ((\phi + \pi) \bmod 2\pi) - \pi.$$

These equations correspond exactly to the update rules applied in the environment: the normalized action is first scaled to (v_i, ω_i) using the bounds above, then the position increment $(\Delta x, \Delta y)$ is computed along the current heading by $(v_i \cos \theta_i, v_i \sin \theta_i)$, and finally the heading is advanced by ω_i and wrapped to the principal interval.

2.2.4. Reward Calculation

The total reward for agent i at time step t is formulated as a weighted sum of active reward components, each representing a distinct behavioral incentive. The scalar reward is

$$R_i(t) = \alpha^\top \mathbf{r}_i(t),$$

where

$$\mathbf{r}_i(t) = [r_i^{\text{prog}}, r_i^{\text{goal}}, r_i^{\text{cover}}, r_i^{\text{coll,a}}, r_i^{\text{coll,o}}, r_i^{\text{lin}}, r_i^{\text{dir}}, r_i^{\text{time}}]^\top.$$

The active reward weights used in the full method and in the ablation experiments are listed in Table 1. The unused zero-weight logging channel is omitted from the scalar reward definition. These weights were selected empirically during preliminary tuning and then kept fixed across the reported team sizes and seeds.

Table 1. Reward-component weights for the full method and ablated variants. The no-assignment-shaping variant removes the progressive and assigned-distance shaping terms. The no-collision variant removes collision-penalty terms.

Component	Full	No-Assignment	No-Collision
Progressive reward, r_i^{prog}	1	0	1
Distance-to-goal penalty, r_i^{goal}	1	0	1
Landmark coverage reward, r_i^{cover}	10	10	10
Agent–agent collision penalty, $r_i^{\text{coll,a}}$	10	10	0
Inactive-agent/obstacle collision penalty, $r_i^{\text{coll,o}}$	10	10	0
Linear velocity reward, r_i^{lin}	1	1	1
Directional alignment reward, r_i^{dir}	1	1	1
Time penalty, r_i^{time}	1	1	1

The weights are therefore treated as fixed experimental hyperparameters rather than values derived from an analytical optimum; the effect of the collision-kernel sharpness parameter is examined in Section 3.12.

Progressive reward measures stepwise improvement towards the assigned landmark. It is defined as the difference between the previous and current assignment distances (obtained via the Hungarian algorithm), normalized by the maximum linear velocity:

$$r_i^{\text{prog}}(t) = \frac{d_i^{\text{old}} - d_i^{\text{new}}}{v_{\max}}.$$

This term is clipped to the range $[-1, 1]$ to avoid extreme fluctuations. It rewards agents for reducing their assignment distance and penalizes them when they move away from their target.

Distance-to-goal penalty provides a dense shaping signal based on the normalized Hungarian assignment distance:

$$r_i^{\text{goal}}(t) = -\frac{d_i^{\text{new}}}{L},$$

where L denotes the environment size. This term penalizes large distances and encourages agents to approach their assigned landmark. Both r_i^{prog} and r_i^{goal} directly rely on the assignment distance, which is determined using the Hungarian algorithm that will be described in the next subsection.

Landmark coverage reward gives a discrete bonus when an agent reaches its assigned landmark:

$$r_i^{\text{cover}}(t) = \begin{cases} 1, & \text{if agent } i \text{ covers its landmark at time } t, \\ 0, & \text{otherwise.} \end{cases}$$

The corresponding weight in Table 1 provides the positive completion bonus.

Collision penalty captures close proximity between agents and occupied entities. For an active agent i , we use the same exponential proximity kernel for active-agent collisions and inactive-agent/obstacle collisions:

$$r_i^{\text{coll},s}(t) = -\sum_{j \in \mathcal{C}_i^s(t)} \exp\left(-\frac{c d_{ij}}{d_{\text{safe}}}\right), \quad s \in \{\text{a}, \text{o}\},$$

where d_{ij} is the Euclidean distance between agent i and entity j , $\mathcal{C}_i^{\text{a}}(t)$ is the set of active agents within the distance threshold d_{safe} , and $\mathcal{C}_i^{\text{o}}(t)$ is the set of inactive completed agents or obstacle entities within the same threshold. The reported experiments do not include external static obstacles, so the obstacle part of $\mathcal{C}_i^{\text{o}}(t)$ is empty in those experiments. The constant $c = 7$ is a fixed collision-kernel sharpness hyperparameter. Only distances smaller than d_{safe} contribute. This term penalizes close proximity between active agents and occupied entities.

Linear velocity reward encourages agents to move with sufficient forward speed. It is defined as

$$r_i^{\text{lin}}(t) = -\left(1 - \frac{|v_i^{\text{lin}}|}{v_{\text{max}}}\right),$$

where v_i^{lin} is the linear velocity of agent i . This provides higher rewards when agents maintain velocities close to their maximum.

Directional alignment reward promotes efficient motion toward the assigned target. Let θ_i be the heading angle of agent i and $\hat{d}_i$ the unit vector pointing from the agent's position to its assigned landmark. The reward is defined as

$$r_i^{\text{dir}}(t) = \cos(\theta_i, \hat{d}_i) - 1,$$

which takes its maximum at perfect alignment (0) and penalizes misaligned headings with negative values.

Time penalty ensures efficiency by discouraging unnecessarily long trajectories:

$$r_i^{\text{time}}(t) = -1.$$

In summary, the progressive and distance-based components encourage steady motion toward the assigned goals, coverage produces a positive signal upon successful completion,

collision penalties penalize close proximity to occupied entities, velocity and alignment terms shape locomotion, and the constant time penalty incentivizes timely coverage.

2.2.5. Hungarian Assignment Distance

At each time step t , the correspondence between the set of currently active agents $\mathcal{A}_t$ and the set of yet-uncovered landmarks $\mathcal{L}_t$ is determined by solving a linear sum assignment problem with the Euclidean distance as the cost. Let $\mathbf{p}_i(t) \in \mathbb{R}^2$ denote the position of agent $i \in \mathcal{A}_t$ and $\ell_j(t) \in \mathbb{R}^2$ the position of landmark $j \in \mathcal{L}_t$. The cost matrix $\mathbf{C}(t) \in \mathbb{R}^{|\mathcal{A}_t| \times |\mathcal{L}_t|}$ is

$$C_{ij}(t) = \|\mathbf{p}_i(t) - \ell_j(t)\|_2, \quad i \in \mathcal{A}_t, j \in \mathcal{L}_t.$$

The optimal one-to-one matching $\sigma_t^* : \mathcal{A}_t \rightarrow \mathcal{L}_t$ is obtained by minimizing the total assignment cost

$$\sigma_t^* = \arg \min_{\sigma \in \mathcal{S}} \sum_{i \in \mathcal{A}_t} C_{i, \sigma(i)}(t),$$

where $\mathcal{S}$ is the set of all injective mappings from $\mathcal{A}_t$ to $\mathcal{L}_t$. The Hungarian distance used by the reward is then the per-agent assignment distance induced by the optimal matching,

$$d_i(t) = C_{i, \sigma_t^*(i)}(t), \quad i \in \mathcal{A}_t,$$

with $d_i(t)$ set to zero for inactive agents (i.e., agents that have already terminated) and for agents without an available landmark when $|\mathcal{L}_t| = 0$. This assignment is recomputed at every step t , so $d_i(t)$ reflects the current geometry and the evolving availability of agents and landmarks. The optimization is solved with the Hungarian method (also known as the Kuhn–Munkres algorithm), which provides an exact polynomial-time solution to the linear assignment problem [69]. For $m = |\mathcal{A}_t| \leq n$ active agents, the assignment step has $O(m^3)$ time complexity and is recomputed at each environment step; this cost is negligible for the evaluated $n \leq 6$ setting but may become relevant for substantially larger teams.

2.3. Proposed Learning Framework

2.3.1. Learning Procedure

The learning procedure employs a shared actor–critic architecture adapted for homogeneous multi-agent systems. A single actor network is shared among all interchangeable agents, leveraging parameter sharing to promote consistent policy behavior across agents. A single critic network is also shared across agents and is applied independently to each local observation–action pair (o_i, a_i) . The critic does not receive the global state or the joint action. This setup uses the symmetry of homogeneous agent roles while preserving decentralized execution [7].

During each learning iteration, a batch of transitions is sampled from the replay buffer, containing current observations, actions taken, decomposed reward components, next observations, and per-agent done flags. With a replay-buffer capacity of 50,000 transitions and a maximum episode length of 500 steps, the buffer can reach capacity after approximately 100 full-length episodes; older transitions are then replaced by newer samples according to the FIFO replay-buffer mechanism. The scalar reward for each agent is obtained from the weighted reward components defined in Section 2.2.4, yielding the per-agent reinforcement signal. The training pipeline also applies offline goal relabeling, following the principle of hindsight experience replay [80], and a fixed outer restart criterion to increase goal-reaching replay transitions and recover persistent no-progress runs; these auxiliary training procedures do not modify the reward definition, actor–critic losses, policy architecture, or deterministic evaluation protocol.

Target Q-values are computed using the target actor and target critic networks. Formally, for agent i in batch entry b , the target is

$$y_i^{(b)} = r_i^{(b)} + \gamma(1 - d_i^{(b)}) Q_{\theta'}(o_i'^{(b)}, \pi_{\phi'}(o_i'^{(b)})),$$

where $r_i^{(b)}$ is the observed reward, $d_i^{(b)}$ is the per-agent done flag, $o_i'^{(b)}$ is the next observation, $\pi_{\phi'}$ is the actor target network, $Q_{\theta'}$ is the critic target network, and γ is the discount factor. Thus, terminated agents do not contribute a bootstrap term. The critic is optimized by minimizing the mean squared error between the predicted Q-values and these targets over valid samples:

$$\mathcal{L}_{\text{critic}} = \frac{1}{|\mathcal{B}|} \sum_{(b,i) \in \mathcal{B}} [Q_{\theta}(o_i^{(b)}, a_i^{(b)}) - y_i^{(b)}]^2,$$

where $\mathcal{B}$ denotes the set of valid (batch, agent) pairs. Terminal reaching transitions are retained in $\mathcal{B}$, while terminated idle samples are masked out from the actor and critic losses.

Actor updates rely on maximizing the expected Q-value under the shared policy. The corresponding loss is defined as the negative expected action-value across all agents:

$$\mathcal{L}_{\text{actor}} = -\frac{1}{|\mathcal{B}|} \sum_{(b,i) \in \mathcal{B}} Q_{\theta}(o_i^{(b)}, \pi_{\phi}(o_i^{(b)})).$$

This formulation allows policy gradients to flow through the critic while ensuring contributions from each agent are aggregated equally.

Soft updates of target network weights are performed via exponential moving averages:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta', \quad \phi' \leftarrow \tau\phi + (1 - \tau)\phi',$$

with $\tau \ll 1$. This strategy stabilizes training by slowly tracking the main network parameters, following the DDPG paradigm [3].

DDPG was selected because the action space consists of continuous velocity commands and because its off-policy actor–critic update is compatible with replay-buffer training. Alternative continuous-control methods such as TD3 and SAC address known stability issues in actor–critic learning, while PPO/MAPPO-style methods provide relevant on-policy cooperative MARL alternatives. The present study does not claim formal convergence guarantees for nonlinear multi-agent DDPG; the evidence is empirical and is based on multi-seed training, ablation, baseline, and generalization results.

By combining a shared actor with a shared local critic, this method aggregates experience across homogeneous agents while preserving decentralized execution at evaluation time.

2.3.2. Shared Actor–Critic Architecture

The concept of a shared actor–critic architecture relies on the principle that agents with identical capabilities and roles can operate using the same neural parameterization. In this framework, both the actor and critic networks are common across all agents rather than individualized per agent. Parameter sharing enforces a common policy and value-function parameterization for homogeneous agents and promotes experience sharing across agents. This property is useful when agents are symmetric in terms of observations, action spaces, and objectives, because it reduces redundant network copies during training.

The actor network, illustrated in Figure 3, takes as input the local observation o_i of agent i . The input is first passed through a linear transformation, followed by layer normalization and a rectified linear unit (ReLU) nonlinearity. The transformed representation is subsequently processed by a second linear layer with another ReLU activation.

The final layer applies a hyperbolic tangent (tanh) activation to constrain the output to the valid action range, resulting in $a_i = \pi_\theta(o_i)$. In the experiments, the actor hidden width is $\max(256, \text{obs_dim})$. This architecture ensures that each agent produces an action based solely on its local observation, without access to observations or actions of other agents, thus respecting decentralization at the execution stage.

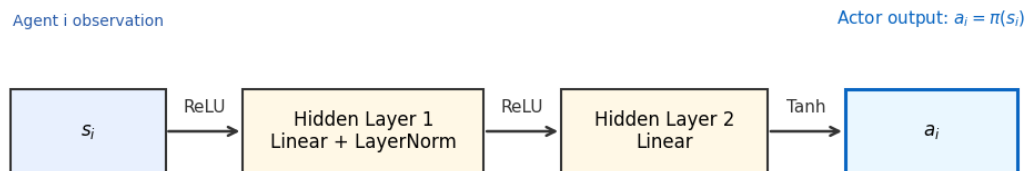


Figure 3. Shared actor network mapping the agent observation o_i to the action $a_i = \pi_\theta(o_i)$.

The critic network, shown in Figure 4, estimates the value of a local observation–action pair by taking as input the concatenation of o_i and a_i . This concatenated vector is processed by a linear transformation, layer normalization, and a ReLU activation, producing an intermediate representation that is passed through a second linear transformation with another ReLU nonlinearity. The final output of the network corresponds to the scalar value $Q_\phi(o_i, a_i)$. The critic hidden width is $\max(128, \text{input_dim})$. Unlike centralized CTDE critics that depend on joint state–action representations, this formulation restricts the input to the local pair (o_i, a_i) . The critic is therefore shared across homogeneous agents but remains local in its input structure; it does not explicitly estimate a joint state–action value or solve multi-agent credit assignment through centralized information.

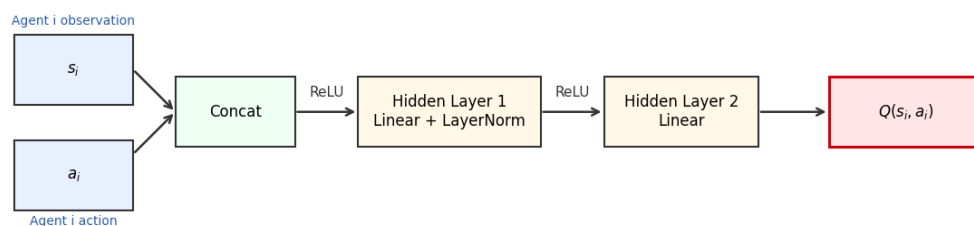


Figure 4. Shared critic network mapping the observation–action pair (o_i, a_i) to the value estimate $Q_\phi(o_i, a_i)$.

Table 2 summarizes the hidden widths and parameter counts used for the shared actor and critic networks in the experiments.

Table 2. Shared actor–critic network dimensions used in the experiments. Parameter counts are reported for all evaluated team sizes. For the critic, the input dimension is the concatenated observation–action dimension.

Network	n	Input Dimension	Hidden Width	Parameters
Actor	4	27	$\max(256, \text{obs_dim})$	73,986
Actor	5	35	$\max(256, \text{obs_dim})$	76,034
Actor	6	43	$\max(256, \text{obs_dim})$	78,082
Critic	4	29	$\max(128, \text{input_dim})$	20,737
Critic	5	37	$\max(128, \text{input_dim})$	21,761
Critic	6	45	$\max(128, \text{input_dim})$	22,785

2.4. Overall Algorithm

The training framework follows an actor–critic paradigm in which the shared actor selects actions from local observations, and the shared critic evaluates local observation–action pairs. Learning proceeds through interaction with the environment and storage of

per-agent experience tuples in a replay buffer. At fixed intervals, mini-batches are sampled and used to update both networks. Target networks are maintained to stabilize updates and are softly updated to avoid oscillations. Algorithm 1 summarizes the main computational steps, including per-agent done masking and valid-sample filtering.

Algorithm 1 Training loop with shared actor–critic learning and per-agent masking

```

1: Initialize environment; replay buffer  $\mathcal{D}$ ; shared actor  $\pi_\theta$ ; shared critic  $Q_\phi$ ; target
   networks  $\pi_{\bar{\theta}}, Q_{\bar{\phi}}$ 
2: for episode = 1 to  $N$  do
3:   Reset environment and obtain local observations  $\{o_i\}_{i=1}^n$ 
4:   while episode not terminated do
5:     for each agent  $i$  do
6:       Select action  $a_i \leftarrow \pi_\theta(o_i)$ 
7:     end for
8:     Execute joint action  $a = \{a_i\}_{i=1}^n$ 
9:     Observe next observations  $\{o'_i\}_{i=1}^n$ , rewards  $\{r_i\}_{i=1}^n$ , and per-agent done flags
        $\{d_i\}_{i=1}^n$ 
10:    Store  $\{(o_i, a_i, r_i, o'_i, d_i)\}_{i=1}^n$  in  $\mathcal{D}$ 
11:    if buffer ready then
12:      Sample minibatch from  $\mathcal{D}$ 
13:      Define valid set  $\mathcal{B}$  by retaining terminal reaching transitions and masking
        terminated idle samples
14:      for each valid pair  $(b, i) \in \mathcal{B}$  do
15:         $a_i^{(b)} \leftarrow \pi_{\bar{\theta}}(o_i^{(b)})$ 
16:         $y_i^{(b)} \leftarrow r_i^{(b)} + \gamma(1 - d_i^{(b)})Q_{\bar{\phi}}(o_i^{(b)}, a_i^{(b)})$ 
17:      end for
18:      Update critic by minimizing
        
$$\frac{1}{|\mathcal{B}|} \sum_{(b,i) \in \mathcal{B}} \left( Q_\phi(o_i^{(b)}, a_i^{(b)}) - y_i^{(b)} \right)^2$$

19:      Update actor by minimizing
        
$$-\frac{1}{|\mathcal{B}|} \sum_{(b,i) \in \mathcal{B}} Q_\phi(o_i^{(b)}, \pi_\theta(o_i^{(b)}))$$

20:       $\bar{\phi} \leftarrow \tau\phi + (1 - \tau)\bar{\phi}$ 
21:       $\bar{\theta} \leftarrow \tau\theta + (1 - \tau)\bar{\theta}$ 
22:    end if
23:     $o_i \leftarrow o'_i$  for all agents
24:  end while
25: end for

```

Table 3 lists the training hyperparameters and reproducibility details used in the experiments. The table also records the collision-kernel sharpness parameter, the number of independent seeds, and the network dimensions requested for reproducibility.

Table 3. Hyperparameters and reproducibility details used in training.

Parameter	Value
Actor learning rate	1×10^{-3}
Critic learning rate	1×10^{-3}
Discount factor γ	0.99
Soft update rate τ	0.005
Batch size	128

Table 3. Cont.

Parameter	Value
Replay buffer size	50,000
Train frequency	every 100 steps
Max episode steps	500
Environment size	20×20
Independent seeds	5
Agent radius	0.5
Safe distance	$2v_{\text{lin,max}}$
Collision-kernel sharpness	$c = 7$
Reward weights	Table 1
Actor hidden width	$\max(256, \text{obs_dim})$
Critic hidden width	$\max(128, \text{input_dim})$
Actor parameters	73,986 ($n = 4$); 76,034 ($n = 5$); 78,082 ($n = 6$)
Critic parameters	20,737 ($n = 4$); 21,761 ($n = 5$); 22,785 ($n = 6$)

3. Results

We evaluated the shared actor–critic policy on cooperative landmark coverage tasks with team sizes $n \in \{4, 5, 6\}$. The main training evaluation uses five independent random seeds for each team size and reward configuration, with 1000 training episodes per run. Unless otherwise stated, final-window metrics are computed over the last 200 training episodes of each run and are reported as mean $\pm$ standard deviation across seeds. These training-window metrics are computed from the training trajectories and therefore include the exploration setting used during training. Deterministic evaluation is reported separately using saved policies without exploration noise and is used for environment-size transfer, completion-time curves, direct collision-rate measurement, and path-length diagnostics. The reported results should therefore be read as fixed-budget empirical performance rather than as evidence of asymptotic convergence.

Let $r_{i,k}(t)$ denote the k -th active reward component for agent i at timestep t , using the reward-component ordering defined in Section 2.2.4. For the full reward configuration, the fixed scaling vector is

$$\alpha = (1, 1, 10, 10, 10, 1, 1, 1).$$

The team return for episode e is

$$R_e = \sum_t \sum_{i=1}^n \sum_{k=1}^8 \alpha_k r_{i,k}(t). \quad (1)$$

An episode is counted as a full-team success only if all n agents reach their assigned landmarks within the horizon. Writing $S_e \in \{0, 1\}$ for this full-team success indicator over a set $\mathcal{E}$ of episodes, the empirical full-team success rate is

$$\hat{p} = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} S_e. \quad (2)$$

Partial task completion is reported separately using the coverage metric

$$\hat{c} = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} \frac{\text{done_count}_e(T_{\max})}{n}, \quad (3)$$

where $T_{\max}$ is the episode horizon. Thus, an episode in which only a subset of agents completes the task contributes zero to full-team success but contributes proportionally to coverage.

For completed episodes, time-to-completion is defined as

$$T_e = \min\{t : \text{done_count}_e(t) = n\}. \quad (4)$$

Episodes without full-team completion before the horizon are retained as non-completions at $T_{\max}$. Completion-time behavior is summarized using empirical full-team completion-time curves, where the curve value at timestep t is the fraction of evaluation episodes that have completed by that timestep.

Uncertainty in the quantitative results is evaluated at the run level rather than by treating episodes from a single training trajectory as independent samples. Final-window full-team success rate, coverage, return, and completion-time summaries are therefore reported across independent seeds. For reward-configuration comparisons, we use seed-level Kruskal–Wallis tests as omnibus rank-based comparisons and Mann–Whitney tests for pairwise contrasts [81,82]. Because each configuration uses five seeds, these tests are interpreted as low-power seed-level checks rather than as equivalence tests; non-significant results are not taken as evidence that two configurations are equivalent or that an ablated component is unnecessary.

The subsections below report multi-seed learning performance, reward decomposition and collision metrics, final-window performance across the evaluated team sizes, assignment-shaping sample efficiency, training-pipeline ablations, diagnostic analysis of the $n = 5$ case, qualitative and path-length diagnostics, seed-level statistical comparisons, contextual positioning relative to prior work, reward ablations and sensitivity checks, within-environment baselines, environment-size transfer, and computational cost. All team-size conclusions are restricted to the evaluated range $n \in \{4, 5, 6\}$, and deterministic-evaluation results are distinguished from training-window results where they are reported.

3.1. Learning Performance

We assess learning using the rolling full-team success rate, averaged over five independent seeds for each team size and reward configuration. We compare the full reward configuration with two ablated variants: no-assignment-shaping removes the progressive and assigned-distance shaping terms, while no-collision removes the collision-penalty terms. These variants are analyzed quantitatively in Sections 3.5, 3.6 and 3.10.

For each run and reward configuration, we compute a rolling success estimate with window $w = 50$,

$$\tilde{p}_e = \frac{1}{w} \sum_{j=\max(1, e-w+1)}^e S_j, \quad (5)$$

where $S_j \in \{0, 1\}$ indicates whether episode j achieved full-team success. The curves are computed separately for each seed and then averaged across the five independent seeds.

Figure 5 shows that the rolling full-team success rate improves during the fixed training budget for all evaluated team sizes and reward configurations. The individual seed traces show that variability differs across settings, especially for some $n = 5$ runs. Therefore, final-window performance is summarized quantitatively in Section 3.5, while assignment-shaping effects on learning dynamics are analyzed separately in Section 3.6 and seed-level statistical comparisons are reported in Section 3.10. The curves are not interpreted as proof of asymptotic convergence.

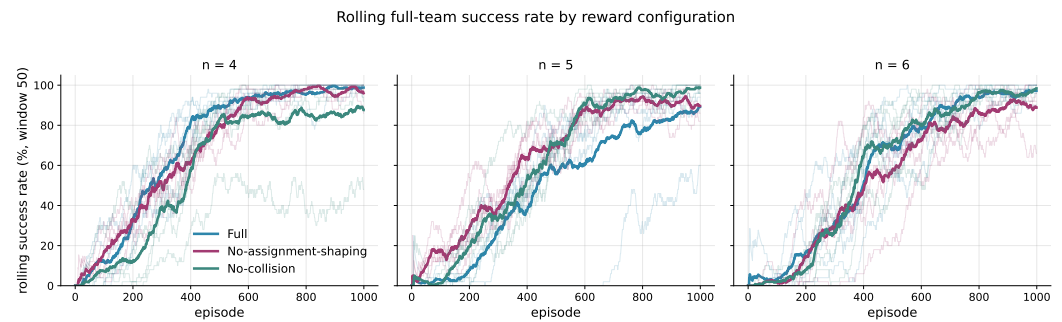


Figure 5. Rolling full-team success rate over training using the estimator in (5). Thick curves show the mean over five independent seeds for each team size and reward configuration; faint curves show individual seed trajectories. The no-assignment-shaping variant removes the progressive and assigned-distance shaping terms, while the no-collision variant removes the collision-penalty terms.

3.2. Coordination Efficiency

We evaluate how rapidly and progressively the teams achieve full-team coverage under the full reward configuration. For completed episodes, T_e denotes the completion time defined in Equation (4). Coordination quality is reflected in the distribution of T_e across successful final-window episodes, the empirical cumulative probability of full-team completion as time advances, and the mean progression of the number of completed agents within an episode. For these diagnostic plots, curves and uncertainty bands are computed at the seed level over five independent runs.

Figure 6 summarizes the final-window completion-time distributions for successful full-team completion episodes. Most successful episodes complete before the 500-step horizon, although the distributions retain long tails. Because this plot is restricted to successful episodes, incomplete episodes are reflected separately in the cumulative completion curves and full-team success rates. Thus, the final trained policies often achieve full-team coverage within the horizon, while completion-time variability remains visible across team sizes.

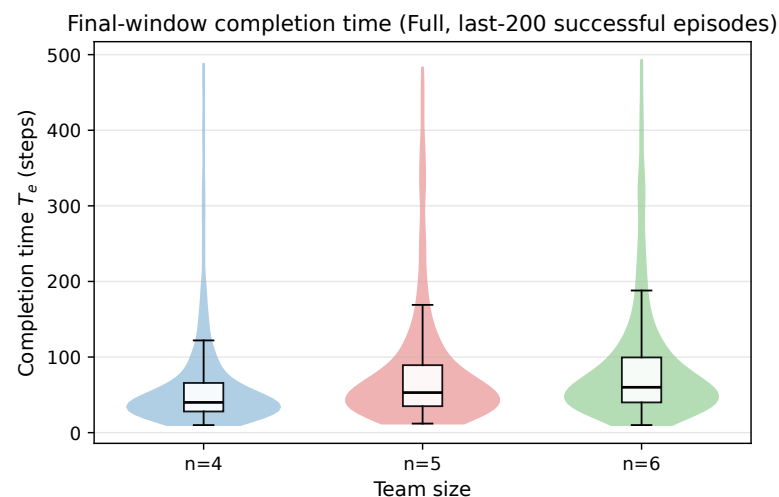


Figure 6. Completion-time distribution T_e for the full reward configuration over the final training window. The distribution is restricted to successful full-team completion episodes from the last 200 episodes of each seed and pooled across five independent seeds.

To connect these distributions to reliability over time, we compute the empirical cumulative full-team completion function separately for each seed,

$$\hat{F}_{n,s}(t) = \frac{1}{|\mathcal{E}_{n,s}|} \sum_{e \in \mathcal{E}_{n,s}} \mathbf{1}\{\text{done_count}_e(t) = n\}, \quad (6)$$

where $\mathcal{E}_{n,s}$ is the final-window episode set for team size n and seed s . This quantity reports, at each timestep, the fraction of episodes in that seed that have already achieved full-team completion. Episodes without full-team completion remain non-completed throughout the horizon.

Figure 7 shows a steep early rise in completion probability, indicating that many final-window episodes complete substantially before the horizon. The $n = 4$ and $n = 6$ curves reach high cumulative completion levels, while the $n = 5$ curve remains lower and shows wider seed-level variability. A faster rise of $\hat{F}_{n,s}(t)$ means that a larger fraction of episodes has already completed at earlier timesteps.

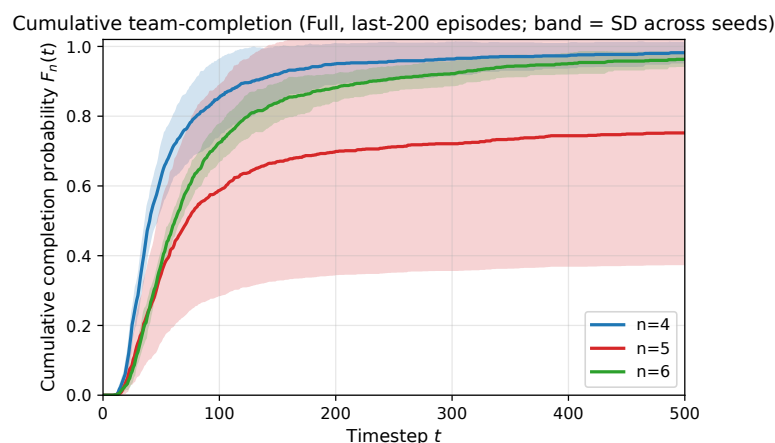


Figure 7. Empirical cumulative full-team completion probability $\hat{F}_{n,s}(t)$ for the full reward configuration over the last 200 episodes. Curves show the mean across five seed-level cumulative functions; shaded regions denote standard deviation across seeds.

Finally, we examine how partial progress evolves within an episode by tracking the mean number of completed agents as a function of time. Writing $\text{done_count}_{e,s}(t)$ for the number of agents that have finished by timestep t in episode e from seed s , with post-completion values carried forward to the horizon, we estimate the seed-level mean

$$\hat{m}_{n,s}(t) = \frac{1}{|\mathcal{E}_{n,s}^W|} \sum_{e \in \mathcal{E}_{n,s}^W} \text{done_count}_{e,s}(t), \quad (7)$$

where $\mathcal{E}_{n,s}^W$ denotes either the early or late episode window for team size n and seed s . The plotted curves show the mean of $\hat{m}_{n,s}(t)$ across seeds, with standard deviation across seeds as the uncertainty band.

Figure 8 shows a pronounced change in intra-episode progress from early to late training. Early curves increase gradually, indicating delayed and partial goal attainment. Late curves rise more rapidly for all evaluated team sizes, showing that multiple agents complete their assigned landmarks earlier in the episode. For $n = 4$ and $n = 6$, the late curves approach the full-team ceiling, whereas $n = 5$ remains lower on average and shows wider seed-level variability. This pattern is consistent with the final-window completion-time distributions in Figure 6 and the cumulative completion dynamics in Figure 7.

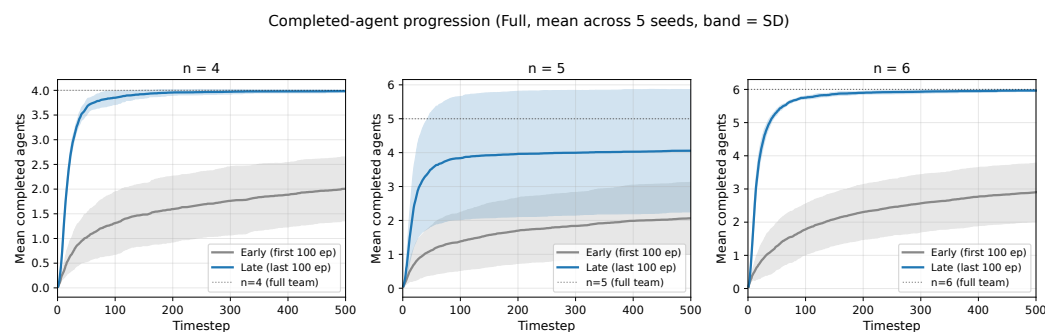


Figure 8. Mean progression of completed agents $\hat{m}_{n,s}(t)$ for the full reward configuration. Early curves use the first 100 episodes, and late curves use the last 100 episodes. Curves show the mean across five seeds; shaded regions denote standard deviation across seeds.

3.3. Reward Decomposition

To interpret how the shaped objective in (1) changes during training under the full reward configuration, we decompose the per-episode return into componentwise contributions using the active reward ordering defined in Table 1. This decomposition is descriptive: it shows how the shaped objective evolves, but it is not interpreted as evidence that any individual component is independently necessary. Let

$$R_{e,k} = \sum_t \sum_{i=1}^n \alpha_k r_{i,k}(t), \quad (8)$$

so that $R_e = \sum_{k=1}^8 R_{e,k}$. We summarize the componentwise values over training episodes and average them across five independent seeds. The scalar-return decomposition uses the active reward components in Table 1; the plotted diagnostic also includes the logged d_{global} channel, which is not included in R_e .

Figure 9 shows that the shaped return is strongly influenced by dense penalty terms, especially distance, directional-alignment, linear-velocity, and time components. As training progresses, progress and landmark-coverage contributions become more evident, while several penalty contributions decrease in magnitude. This pattern is consistent with improved final-window coverage and completion behavior, but it should be interpreted as a diagnostic of training dynamics rather than as evidence that the full reward configuration is statistically superior to the ablated configurations.

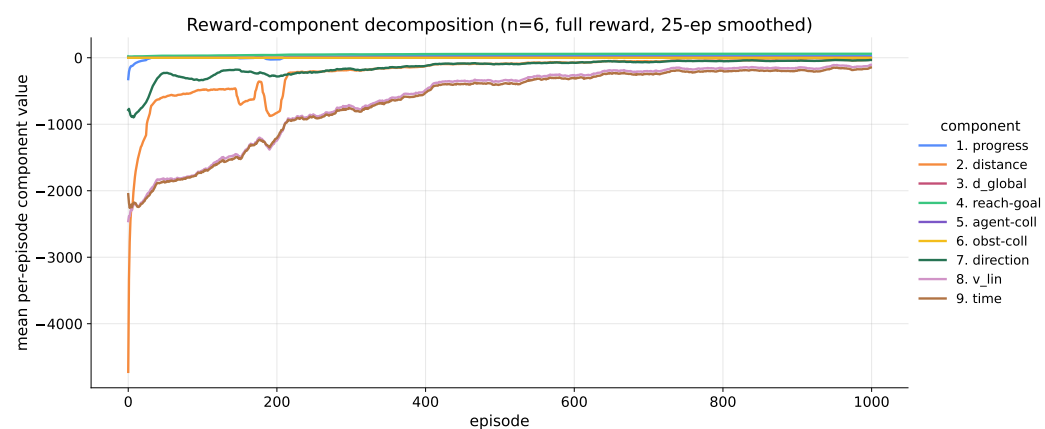


Figure 9. Componentwise return decomposition for the full reward configuration. Curves show smoothed per-episode scaled contributions of the logged reward components. The d_{global} channel is shown only as a diagnostic logging quantity and is not included in the scalar reward R_e . The figure is used as a diagnostic of how the shaped objective evolves during training, not as an ablation of component necessity.

To examine collision-related behavior more directly, we report both the shaped agent–agent collision penalty and a direct collision-count metric. For each training episode, we sum the scaled agent–agent collision component over agents and timesteps and track its magnitude through training. For deterministic evaluation, we also measure collision steps directly: a collision step is counted when at least one active agent–agent collision is recorded at that timestep.

Figure 10 displays an early increase in agent–agent collision penalty magnitude followed by a decline as training advances. The transient increase is consistent with exploratory policies that occasionally choose conflicting motions; the subsequent reduction indicates fewer collision-prone interactions in the simulated training setting. To complement the shaped penalty with a direct metric, the deterministic evaluation measured the collision-step rate. In the 20×20 training arena, the mean collision-step rates were 1.56%, 1.62%, and 2.63% for $n = 4$, $n = 5$, and $n = 6$, respectively. In the 25×25 transfer arena, the corresponding rates were 1.27%, 1.54%, and 1.85%. Thus, collisions remain measurable but relatively infrequent on a per-step basis, while the rate increases with team size in the training arena.

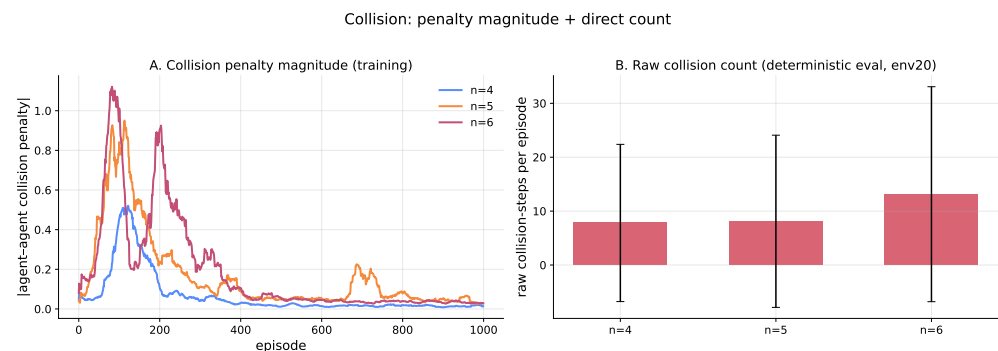


Figure 10. Collision-related diagnostics for the full reward configuration. Panel (A) shows the per-episode agent–agent collision-penalty magnitude during training. Panel (B) reports the raw collision-step count per episode under deterministic evaluation in the 20×20 training arena.

Taken together, Figures 9 and 10 indicate that return improvements are associated with changes across several shaped components rather than with a single reward term. These collision-related results should be interpreted as simulation-level evidence of reduced collision-prone behavior, not as a formal safety guarantee.

3.4. Task-Specific Behavior

We probe *how* the policy produces the observed coverage behavior under the full reward configuration by examining geometric and kinematic traces. Let $x_i(t) \in \mathbb{R}^2$ denote the position of agent i at timestep t , and let $\{g_j\}_{j=1}^n$ be the landmark set. The instantaneous Hungarian (assignment) distance is the minimum average agent–goal distance under a one-to-one matching,

$$D(t) = \min_{\sigma \in S_n} \frac{1}{n} \sum_{i=1}^n \|x_i(t) - g_{\sigma(i)}\|_2, \quad (9)$$

where S_n is the set of permutations of $\{1, \dots, n\}$. The minimization is solved at each t with the Hungarian (Kuhn–Munkres) algorithm [69,70]. For the diagnostic plots in this section, early and late curves are computed separately for each seed and then summarized across five independent seeds.

Figure 11 shows a rapid initial drop of $D(t)$ in the late phase for all evaluated team sizes. The late curves remain below the corresponding early curves, indicating faster reduction of assignment distance after training. The effect is strongest for $n = 4$ and $n = 6$, whereas $n = 5$ shows wider seed-level variability. These traces indicate that the learned

policy more quickly reduces assignment distance in the final training phase, although the degree of consistency differs across team sizes.

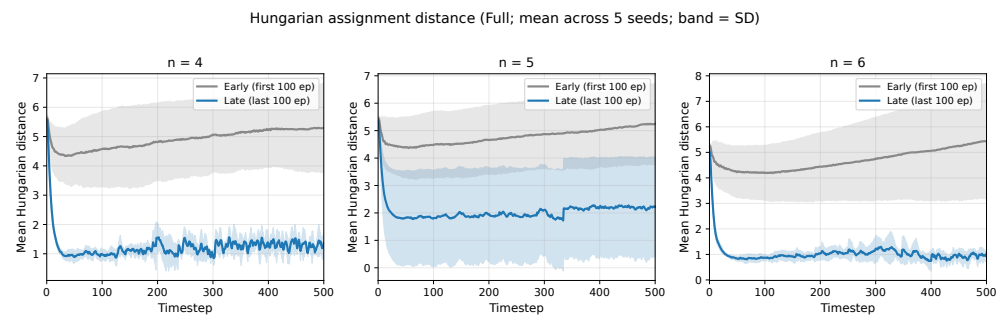


Figure 11. Mean Hungarian distance $D(t)$ versus timestep for the full reward configuration. Early curves use the first 100 episodes, and late curves use the last 100 episodes. Curves show the mean across five seeds; shaded regions denote standard deviation across seeds.

Kinematic profiles complement the geometric picture. Writing $v_i(t)$ for the linear speed and $\omega_i(t)$ for the angular speed (signed turn rate) of agent i , we summarize per-timestep means

$$\bar{v}(t) = \frac{1}{n} \sum_{i=1}^n v_i(t), \quad \bar{\omega}(t) = \frac{1}{n} \sum_{i=1}^n |\omega_i(t)|. \quad (10)$$

These quantities reflect how aggressively the team moves and how much heading correction is required.

Figure 12 shows a burst-settle pattern in the late phase. The late curves exhibit a short initial velocity transient, followed by lower sustained linear and angular velocities. The reduced angular activity is consistent with fewer heading corrections after the initial approach phase and aligns with the lower late-phase Hungarian distances in Figure 11. Early curves maintain higher linear and angular velocities for longer, indicating less settled motion during the initial training phase. Together, these measurements link the observed co-ordination behavior to faster assignment-distance reduction, earlier approach to landmarks, and reduced heading correction in the final training phase.

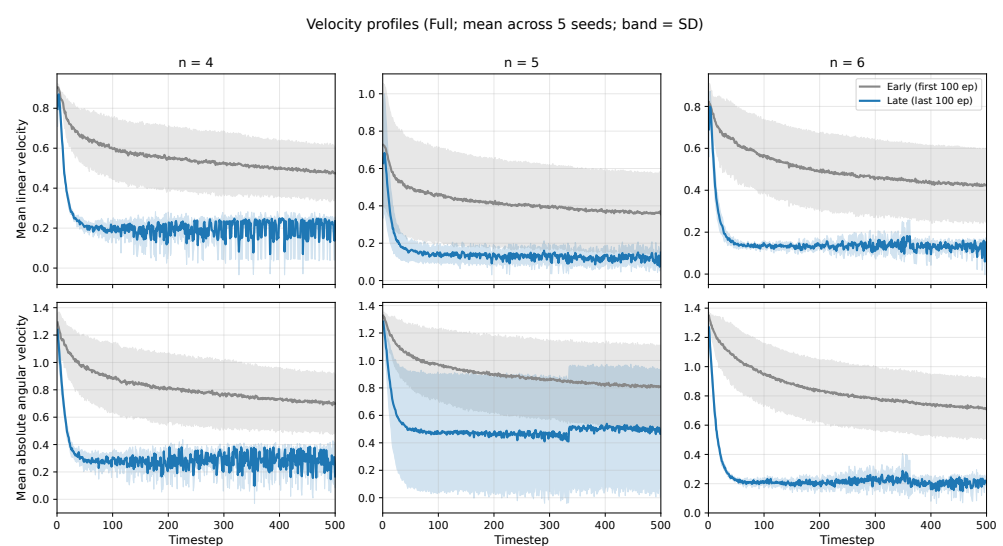


Figure 12. Velocity profiles for the full reward configuration. (**Top** row): mean linear velocity $\bar{v}(t)$. (**Bottom** row): mean absolute angular velocity $\bar{\omega}(t)$. Early curves use the first 100 episodes, and late curves use the last 100 episodes. Curves show the mean across five seeds; shaded regions denote standard deviation across seeds.

3.5. Performance Across Evaluated Team Sizes

We summarize final-window performance across the evaluated team sizes $n \in \{4, 5, 6\}$ using five independent seeds per reward configuration. The final window is the last 200 training episodes of each run. Full-team success is counted only when *all* n agents reach their goals within the horizon; time-limit terminations without full coverage are not successes. Partial task completion is reported separately using the coverage metric. Table 4 reports seed-level mean $\pm$ standard deviation for full-team success rate, landmark coverage, and per-episode return under the scaled reward of (1). The reward configurations are defined in Section 3.1; their learning-speed differences are analyzed separately in Section 3.6.

The full reward configuration maintains high average full-team success over the evaluated range, with coverage above 96% in all three team sizes. The no-assignment-shaping and no-collision variants also reach high final-window performance, and the no-collision variant matches or exceeds the full configuration in final-window success for $n = 5$ and $n = 6$; therefore, the reward-shaping terms are not interpreted as producing a final-window success advantage from Table 4 alone. Their effect is instead evaluated through the sample-efficiency analysis in Section 3.6 and the seed-level tests in Section 3.10.

To visualize final-window reliability across the evaluated team sizes and reward configurations, we plot the seed-level full-team success rates with standard-deviation error bars.

Figure 13 summarizes the final-window full-team success statistics graphically and shows the seed-level values behind the aggregate bars. Together with Table 4, the results show high average full-team success and coverage over the evaluated team sizes, while also exposing seed-level variability in some configurations, especially for the full configuration at $n = 5$. These results support conclusions only over the tested range $n \in \{4, 5, 6\}$; larger-team scaling is not inferred from this experiment.

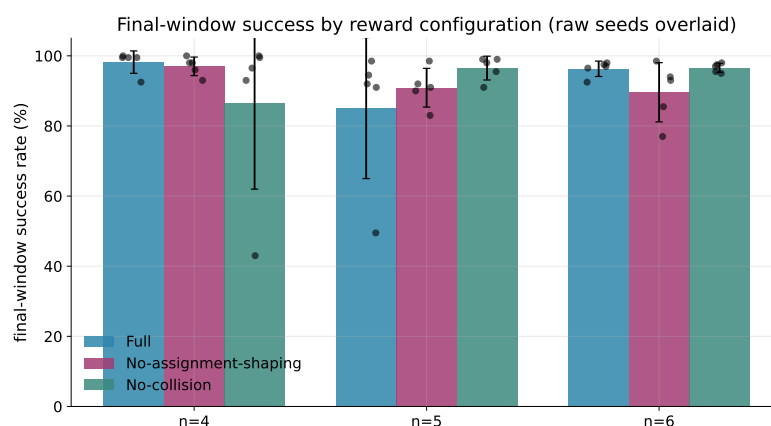


Figure 13. Final-window full-team success rate across the evaluated team sizes and reward configurations. Bars show the mean over five independent seeds; error bars denote standard deviation across seeds; overlaid points show individual seed-level success rates.

Table 4. Final-window summary over five independent seeds. Metrics are computed over the last 200 training episodes of each run and reported as mean $\pm$ standard deviation across seeds. SR: full-team success rate; Coverage: mean fraction of landmarks covered; R: per-episode return under (1).

n	Configuration	SR (%)	Coverage (%)	Mean R
4	Full	98.2 ± 2.9	99.6 ± 0.7	181.0 ± 62.8
4	No-assignment-shaping	97.0 ± 2.4	99.2 ± 0.6	144.4 ± 41.3
4	No-collision	86.4 ± 21.8	96.4 ± 5.8	-21.9 ± 387.0

Table 4. Cont.

<i>n</i>	Configuration	SR (%)	Coverage (%)	Mean <i>R</i>
5	Full	85.1 ± 18.0	96.7 ± 4.1	−32.1 ± 329.7
5	No-assignment-shaping	90.9 ± 4.9	98.2 ± 1.0	102.7 ± 74.4
5	No-collision	96.5 ± 3.0	99.3 ± 0.6	169.7 ± 62.6
6	Full	96.3 ± 2.0	99.4 ± 0.3	216.8 ± 43.8
6	No-assignment-shaping	89.6 ± 7.6	98.2 ± 1.3	91.3 ± 128.1
6	No-collision	96.6 ± 1.2	99.4 ± 0.2	224.1 ± 18.7

3.6. Assignment-Shaping Sample Efficiency

The final-window summaries in Section 3.5 show that the ablated reward configurations can also reach high success after training. Therefore, we additionally evaluate whether the assignment-aware shaping terms affect the speed at which successful coordination emerges under the fixed training budget. For each seed, reward configuration, and team size, we define the first episode at which the rolling success estimate in (5) reaches a threshold of 80%:

$$E_{80} = \min\{e : \bar{p}_e \geq 0.8\}. \quad (11)$$

If the threshold is not reached, the run is treated as not reaching the competence threshold within the training horizon.

Figure 14 shows the seed-level threshold episode E_{80} for each team size and reward configuration. The median threshold episodes are 378, 511, and 453 for the full configuration at $n = 4, 5, 6$, respectively, compared with 491, 544, and 556 for the no-assignment-shaping variant and 423, 548, and 465 for the no-collision variant. The full configuration reaches the threshold earlier in median terms, but the seed-level statistical comparison supports this advantage only for $n = 4$ ($p = 0.028$); the gaps for $n = 5$ and $n = 6$ are not statistically significant at five seeds. Thus, the assignment-aware terms are not interpreted as producing a statistically established final-window success advantage, and the evidence for faster learning is limited to the tested fixed-budget protocol, with the clearest support at $n = 4$.

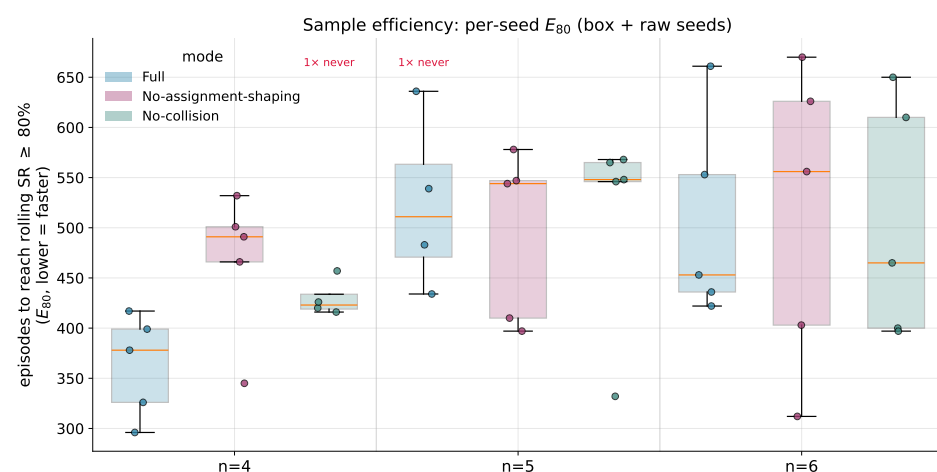


Figure 14. Sample-efficiency comparison across reward configurations using the per-seed threshold episode E_{80} . Lower values indicate earlier emergence of high-rolling full-team success. Boxes summarize the seed-level distributions; overlaid points show individual seeds, and runs that do not reach the threshold within the training horizon are marked separately.

3.7. Training-Pipeline Ablation

The results above evaluate reward-level ablations. To separate these from training-pipeline effects, we additionally ablate two auxiliary training mechanisms used in the full

configuration: offline goal relabeling and orbit-restart handling. The full configuration uses both mechanisms. Since the restart mechanism is implemented on top of the relabeling procedure, the feasible ablation is a three-level ladder: full, no-restart, and no-HER. The no-restart condition keeps goal relabeling but disables orbit-restart handling, while the no-HER condition disables both.

Table 5 and Figure 15 show that removing the restart mechanism substantially increases seed-level variability, especially for $n = 5$ and $n = 6$. In these settings, several no-restart runs collapse to very low success, while other seeds remain competitive. This indicates that the restart mechanism mainly acts as a robustness component for seed-specific orbit or stalling failures, rather than uniformly improving all seeds.

Table 5. Training-pipeline ablation over five independent seeds. Values report final-window full-team success rate (%) over the last 200 training episodes.

n	Condition	HER	Restart	SR (%)
4	Full	Yes	Yes	98.2 ± 2.9
4	No-restart	Yes	No	78.1 ± 39.1
4	No-HER	No	No	96.1 ± 2.7
5	Full	Yes	Yes	85.1 ± 18.0
5	No-restart	Yes	No	23.3 ± 36.6
5	No-HER	No	No	55.5 ± 45.2
6	Full	Yes	Yes	96.3 ± 2.0
6	No-restart	Yes	No	63.9 ± 28.0
6	No-HER	No	No	67.4 ± 22.1

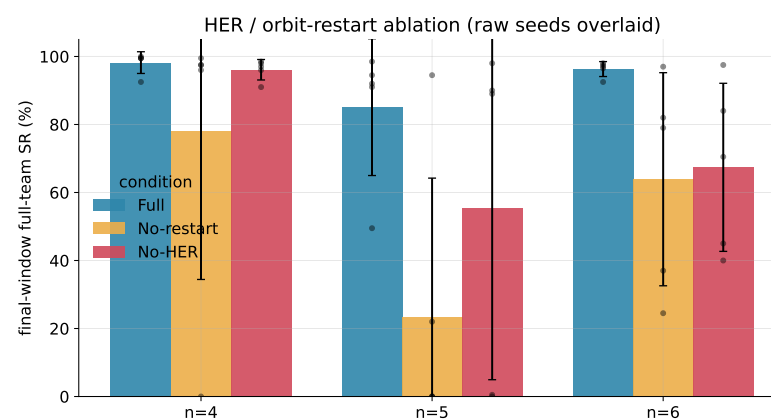


Figure 15. Training-pipeline ablation for HER and orbit-restart handling. Bars show mean final-window full-team success rate over five independent seeds; overlaid points show individual seed-level success rates.

The no-HER condition does not show a uniform degradation at $n = 4$, but it remains less stable at $n = 5$ and $n = 6$. Therefore, the auxiliary training mechanisms should not be interpreted as independently sufficient or uniformly necessary from this ablation alone. Instead, the results support a narrower conclusion: the reported method is an integrated training pipeline in which the reward design provides the main learning signal, while restart handling improves robustness against seed-specific failure modes in the harder team-size settings.

3.8. Diagnostic Analysis of the $N = 5$ Case

The final-window results in Section 3.5 show that the full reward configuration has lower mean success and higher seed-level variability for $n = 5$ than for $n = 4$ and $n = 6$.

To examine whether this reflects a systematic team-size effect or seed-level instability, we inspected the per-seed final-window success rates for the full reward configuration.

Figure 16 shows that the lower average success at $n = 5$ is mainly caused by one seed with 49.5% final-window success. The five $n = 5$ seed values are 49.5%, 94.5%, 98.5%, 92.0%, and 91.0%. Including all seeds gives the reported $85.1 \pm 18.0\%$ success rate. For diagnostic interpretation only, excluding the single low-success seed gives an average of 94.0% with a standard deviation of 2.9% for the remaining four $n = 5$ seeds, which is comparable to the variability observed for $n = 4$ and $n = 6$. All formal summaries and comparisons, however, retain the low-success seed.

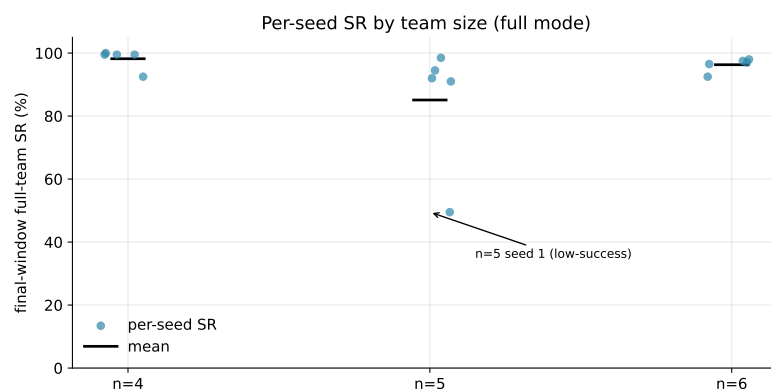


Figure 16. Per-seed final-window success rates for the full reward configuration. The increased variability at $n = 5$ is dominated by one low-success seed, while the remaining $n = 5$ seeds are comparable to the $n = 4$ and $n = 6$ settings. All aggregate results reported elsewhere retain all five seeds.

This diagnostic indicates that the $n = 5$ underperformance is primarily associated with a seed-sensitive failure mode rather than a consistent odd-team-size degradation. In the low-success seed, trajectories and completion statistics indicate an orbiting or stalling pattern in which the policy often reaches partial coverage but does not reliably complete the final assignment within the horizon. This interpretation is consistent with the training-pipeline ablation in Section 3.7, where removing restart handling produced seed-specific collapses rather than uniform degradation across all seeds. The present data therefore identify the dominant empirical source of the $n = 5$ variability, while a full causal analysis of this failure mode remains outside the scope of the current evaluation.

3.9. Qualitative Evaluation

To complement the quantitative analyses, we visualize representative multiagent trajectories from early training and from the final phase for each team size. Colored polylines depict the paths of individual agents within a single episode; blue and green dots mark each agent's start and final positions, respectively, and red crosses ($\times$) mark the landmark (target) positions. Let $x_i(t) \in \mathbb{R}^2$ denote agent i 's position at timestep t . A simple geometric proxy for motion efficiency is the total path length over the evaluated rollout,

$$L_e = \sum_{i=1}^n \sum_{t=0}^{T_e-1} \|x_i(t+1) - x_i(t)\|_2, \quad (12)$$

which decreases when agents move more directly, for a fixed task geometry and evaluation horizon. Table 6 reports this metric for deterministic evaluation, including a normalized ratio relative to the heuristic path length in the 20×20 training arena. The trajectory plots are then used to illustrate path tortuosity, overshoot, and oscillatory motion patterns that are not fully captured by the aggregate scalar metric.

Table 6. Mean total path length L_e in deterministic evaluation. The heuristic controller is included as a non-learning reference in the 20×20 training arena. The ratio L_e/L_{heur} normalizes the learned-policy path length in the training arena by the corresponding heuristic path length. Learned-policy values are also reported for the larger 25×25 arena.

n	Heuristic 20×20	Learned 20×20	L_e/L_{heur}	Learned 25×25
4	27.3	124.0	4.54	212.9
5	33.8	366.1	10.83	533.1
6	39.1	186.8	4.78	345.7

Figure 17 illustrates the qualitative shift in motion patterns for $n = 4$. The early episode displays meandering behavior and wide excursions, while the final-phase episode shows compact paths with smaller terminal excursions. This visual trend is consistent with the lower late-phase Hungarian distances and reduced angular activity observed in Figures 11 and 12, while Table 6 reports the corresponding aggregate path-length metric for deterministic evaluation.

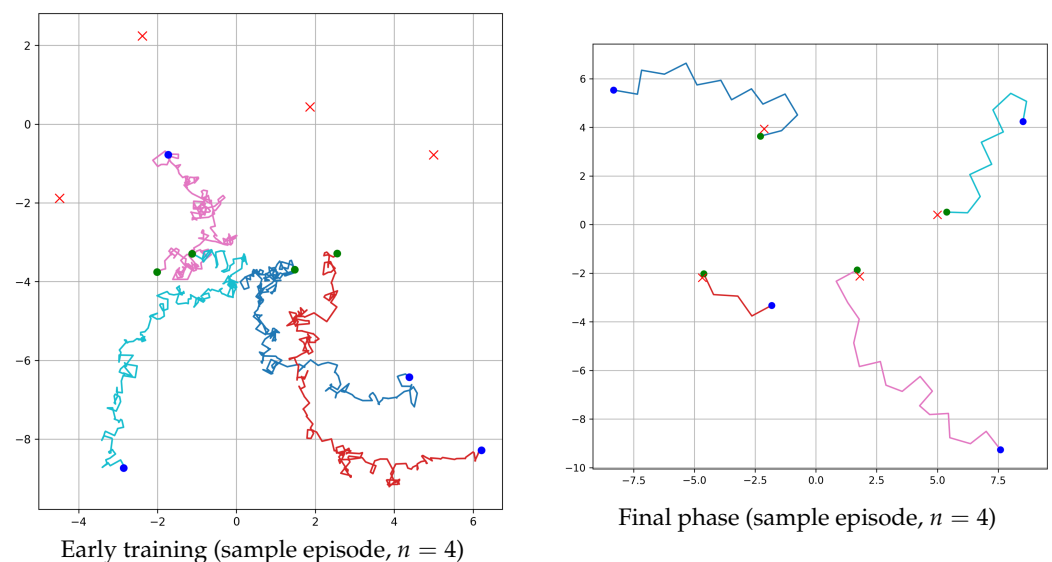


Figure 17. Representative trajectories for $n = 4$. Each coloured polyline is one agent's trajectory (one colour per agent); a blue dot marks that agent's start position and a green dot its final position, and red crosses ($\times$) mark the landmark (target) positions. Early training exhibits longer, more circuitous paths with multiple turns and self-crossings, whereas the final phase shows more decisive, shorter paths with visibly fewer heading reversals.

For $n = 5$ (Figure 18), the early episode shows prolonged looping and broad sweeps, suggesting delayed assignment and frequent heading corrections. By the final phase, paths become straighter with fewer crossings, aligning with the cumulative completion behavior in Figure 7. The larger learned-policy path length and normalized path-length ratio for $n = 5$ in Table 6 are consistent with the higher variability observed for this team size.

The $n = 6$ examples in Figure 19 show a pronounced visual improvement: early trajectories contain dense knots and extended detours, while final-phase paths are cleaner and more targeted. Across $n \in \{4, 5, 6\}$, these qualitative patterns echo the quantitative findings—lower assignment distance, reduced angular adjustments, faster completion, and the raw and normalized path-length values reported in Table 6. The learned policy therefore exhibits a more decisive approach pattern in the final phase, characterized by earlier commitment to an assignment followed by motion with fewer corrective turns.

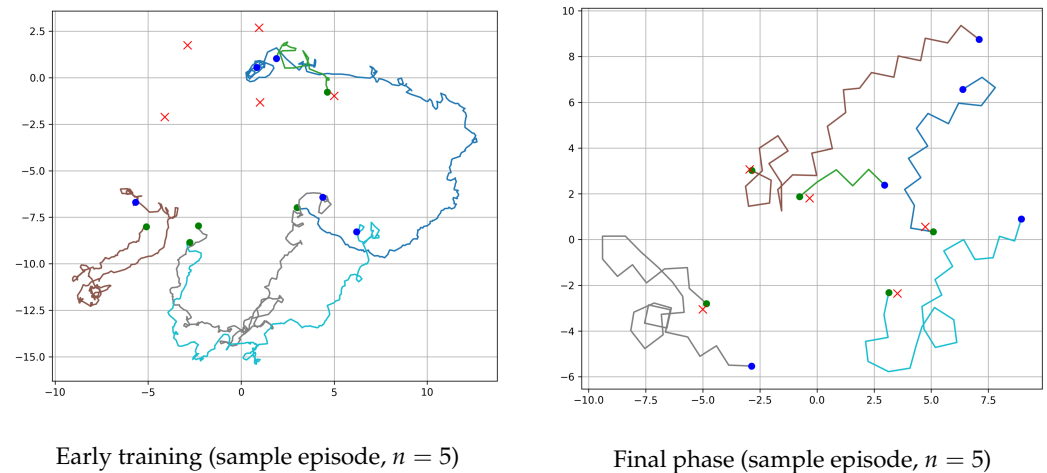


Figure 18. Representative trajectories for $n = 5$. Early paths contain extended loops and wider exploration, whereas late-phase paths are more direct and compact. Colours and symbols are as in Figure 17.

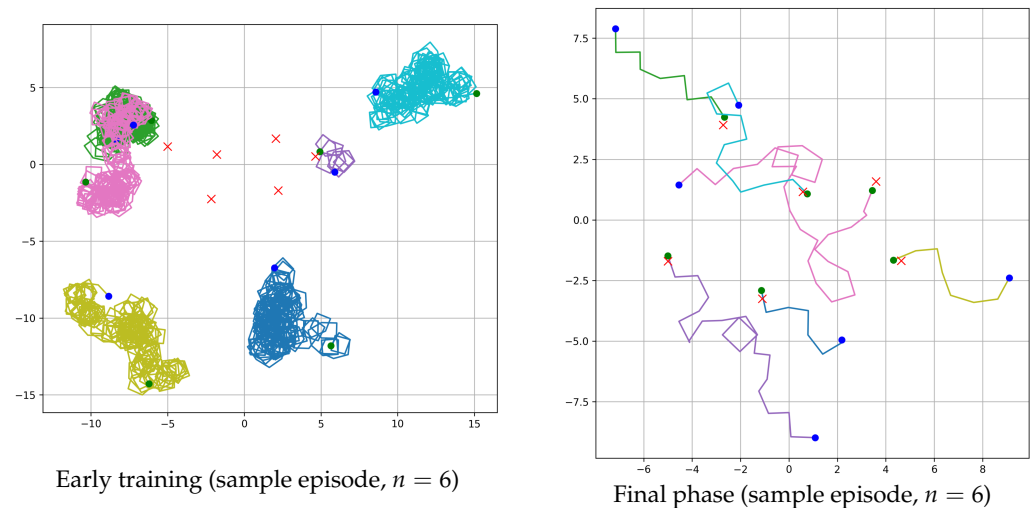


Figure 19. Representative trajectories for $n = 6$. The early episode exhibits clusters of small oscillatory turns and long excursions; the final phase shows markedly cleaner, shorter routes with limited overshoot. Colours and symbols are as in Figure 17.

3.10. Seed-Level Statistical Comparison

The preceding final-window summaries are based on independent training seeds rather than treating episodes from a single run as independent samples. We therefore apply statistical tests at the seed level. For reward-configuration comparisons, the tested samples are the final-window full-team success rates obtained from the five independent seeds for each configuration. We use Kruskal–Wallis tests as omnibus rank-based comparisons across the full, no-assignment-shaping, and no-collision configurations for each team size, followed by Mann–Whitney tests for pooled full-vs-ablation contrasts [81,82]. Because each configuration contains only five seeds, these tests are interpreted as low-power seed-level checks rather than as equivalence tests.

Table 7 summarizes the seed-level statistical comparisons. The Kruskal–Wallis tests do not show significant differences among the three reward configurations for any evaluated team size. However, the post hoc power estimates are low, so the non-significant results should not be interpreted as evidence that the configurations are equivalent or that the ablated components are unnecessary. The pooled Mann–Whitney tests likewise do not show a significant final-window full-team success advantage of the full configuration over either ablated configuration. These tests therefore concern fixed-budget final-window success,

while the learning-speed effect of assignment-aware shaping is evaluated separately in Section 3.6.

The estimated post hoc powers for the Kruskal–Wallis comparisons were 0.07, 0.09, and 0.33 for $n = 4$, $n = 5$, and $n = 6$, respectively, confirming that the non-significant tests should not be interpreted as equivalence.

Table 7. Seed-level statistical comparison of reward configurations. KW compares full, no-assignment-shaping, and no-collision within each team size. MW compares the full configuration against each ablation using pooled seed-level final-window full-team success rates.

Test	Group	Statistic	p	Effect
KW	$n = 4$	$H = 0.90$	0.64	$\epsilon^2 = 0.00$
KW	$n = 5$	$H = 3.54$	0.17	$\epsilon^2 = 0.13$
KW	$n = 6$	$H = 2.57$	0.28	$\epsilon^2 = 0.05$
MW	Full vs. no-collision	$U = 110$	0.93	–
MW	Full vs. no-assignment	$U = 141.5$	0.24	–

Here, KW denotes the Kruskal–Wallis test, and MW denotes the Mann–Whitney test.

Figure 20 summarizes time-to-full-coverage behavior under deterministic evaluation in the 20×20 training arena. A faster rise in the empirical completion curve means that a larger fraction of episodes completes earlier. The $n = 4$ curve rises fastest, while the $n = 5$ curve remains lower because fewer episodes reach full-team completion within the horizon. Pairwise log-rank comparisons of the deterministic completion-time distributions show significant differences among the evaluated team sizes: $n = 4$ vs. $n = 5$, $\chi^2 = 256.99$, $p = 7.8 \times 10^{-58}$; $n = 5$ vs. $n = 6$, $\chi^2 = 99.73$, $p = 1.7 \times 10^{-23}$; and $n = 4$ vs. $n = 6$, $\chi^2 = 56.65$, $p = 5.2 \times 10^{-14}$.

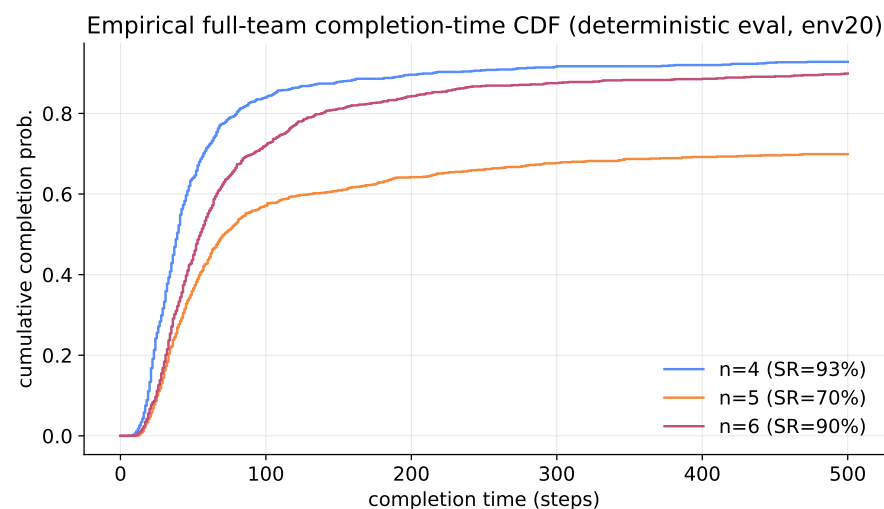


Figure 20. Empirical full-team completion-time cumulative distribution under deterministic evaluation in the 20×20 training arena. The curve value at timestep t is the fraction of evaluation episodes that have completed by that timestep; episodes without full-team completion remain below the final cumulative level.

Overall, the seed-level tests do not support a claim that the full reward configuration is statistically superior to the ablated configurations in final-window full-team success. Because the five-seed tests have low power, this result is interpreted cautiously and is not treated as evidence of equivalence among reward configurations. The final-window results therefore support a limited interpretation: high success and coverage are observed across the evaluated configurations, as summarized in Section 3.5, while the measured

learning-speed advantage of assignment-aware shaping is clearest for $n = 4$, as reported in Section 3.6.

3.11. Contextual Positioning Relative to Prior Work

We situate the evaluated task relative to the recent literature but do not use cross-paper numbers as evidence of relative performance. Environments, horizons, collision models, dynamics, and success definitions differ across studies; therefore, the references in this section are used only for contextual positioning. Direct within-environment comparison against MADDPG and MAPPO is provided in Section 3.13.

CRAMP evaluates grid-based multi-agent path finding on 80×80 maps with obstacle density 0.1 and reports success rate and makespan under its own discrete MAPF protocol [59]. This differs from the continuous-control, obstacle-free coverage task considered here, so CRAMP is used only to contextualize related evaluation metrics rather than to support a direct performance comparison. No MAPF-based baseline is included in the present experiments; hybrid MAPF–reinforcement-learning comparisons are left for future work.

MASP evaluates cooperative navigation in the Multi-Agent Particle Environment and reports success and collision-rate metrics under that benchmark’s dynamics, horizon, and collision accounting [78]. Although Section 3.3 now reports a direct collision-step rate for the present task, its definition, dynamics, and evaluation protocol differ from MASP; therefore, MASP is likewise retained only as contextual background.

Accordingly, no superiority claim is drawn from the CRAMP or MASP references. The quantitative same-environment comparisons are reported in Section 3.13, while the multi-seed final-window performance of the proposed method is reported in Section 3.5. Evaluation on standard MARL benchmarks such as MPE-, VMAS-, or SMAC-style tasks, and direct hybrid MAPF–RL benchmarking, is outside the scope of the present coverage study.

3.12. Reward Ablation and Collision-Parameter Sensitivity

The ablation study uses the same five-seed protocol as the main final-window evaluation. In addition to the no-assignment-shaping variant, which removes the progressive and assigned-distance shaping terms, we also evaluate a no-collision variant, which removes the collision-penalty terms. The reward-component definitions and weights for these configurations are given in Table 1. The reward weights were selected empirically during preliminary tuning and then kept fixed across the reported team sizes and seeds. The analysis below therefore provides a limited empirical sensitivity check through reward-mode ablations and the collision-kernel sweep, rather than a full grid search over all reward weights.

The final-window ablation results are reported in Table 4 and visualized in Figure 13. Across the evaluated team sizes, both ablated variants remain viable in the final training window. In particular, the no-assignment-shaping variant no longer shows a consistent collapse at $n = 6$ under the five-seed evaluation, reaching $89.6 \pm 7.6\%$ final-window success. The no-collision variant reaches $86.4 \pm 21.8\%$, $96.5 \pm 3.0\%$, and $96.6 \pm 1.2\%$ success for $n = 4, 5, 6$, respectively.

The seed-level statistical comparison in Section 3.10 does not show a significant final-window advantage of the full reward configuration over either ablated configuration. Accordingly, assignment-aware and collision-aware shaping are not interpreted as statistically necessary for fixed-budget final-window full-team success in the evaluated obstacle-free setting. This is particularly important for the no-collision variant, which matches or exceeds the full configuration in final-window success for $n = 5$ and $n = 6$.

The main measured effect of assignment-aware shaping is instead the faster emergence of high rolling success reported in Section 3.6, with the clearest seed-level support at $n = 4$.

We further examine the collision-kernel sharpness parameter c , which is fixed to $c = 7$ in the reported experiments. This parameter controls how rapidly the exponential collision penalty decreases with distance. It is treated as an empirical hyperparameter rather than a value derived from first principles. The direct collision-step rates reported in Section 3.3 should therefore be interpreted as simulation-level diagnostics, not as formal safety guarantees.

Figure 21 shows that the converged score is nearly unchanged around the selected value for $n = 4$ and $n = 5$. For $n = 6$, the larger value $c = 15$ produces a lower and more variable score, so it is not used as the default setting. Overall, the sweep supports $c = 7$ as a stable empirical choice for the tested configurations, without implying that it is theoretically optimal. A broader sensitivity study over all reward weights remains outside the scope of the present revision.

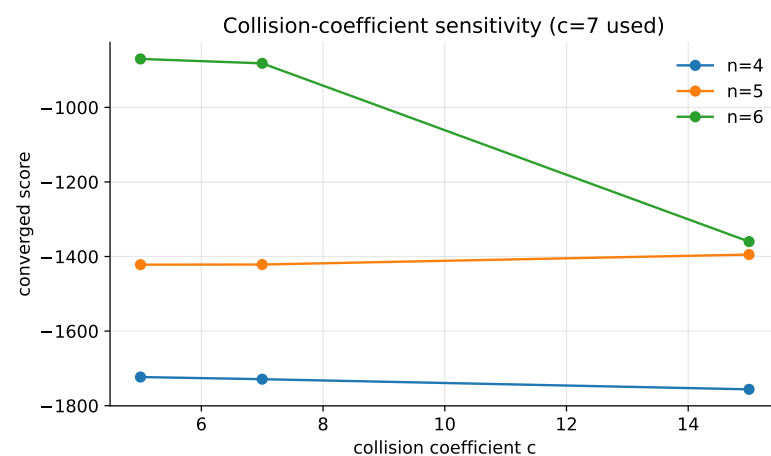


Figure 21. Sensitivity of the converged score to the collision-kernel sharpness parameter c . The reported experiments use $c = 7$. This sweep is a local empirical sensitivity check for the collision kernel, not a full search over all reward weights.

3.13. Within-Environment Baselines

To address the absence of direct within-environment baselines, we compared the proposed full reward configuration with two MARL baselines under the same cooperative coverage setting: MADDPG [31] and MAPPO [36]. The comparison uses the same team sizes, environment size, training horizon, and final-window definition as Section 3.5. Table 8 reports final-window full-team success, while Figure 22 visualizes the corresponding seed-level success rates.

Table 8. Within-environment baseline comparison for final-window full-team success. Metrics are computed over the final training window and reported as mean $\pm$ standard deviation across five independent seeds for each method.

n	Proposed SR (%)	MADDPG SR (%)	MAPPO SR (%)
4	98.2 $\pm$ 2.9	8.9 $\pm$ 10.9	37.0 $\pm$ 39.1
5	85.1 $\pm$ 18.0	0.0 $\pm$ 0.0	16.0 $\pm$ 12.0
6	96.3 $\pm$ 2.0	6.4 $\pm$ 7.8	20.2 $\pm$ 20.8

As shown in Table 8 and Figure 22, the proposed shared actor–critic configuration achieves higher final-window full-team success than both MADDPG and MAPPO in the tested setting. With five seeds per method, one-sided Mann–Whitney tests comparing the proposed method against MAPPO give $p = 0.010$, 0.004 , and 0.006 for $n = 4, 5, 6$,

respectively, with rank-biserial effects of 0.92, 1.00, and 1.00. The corresponding tests against MADDPG give $p = 0.005$, 0.004, and 0.006, with rank-biserial effect 1.00 for all three team sizes. MAPPO improves over MADDPG for full-team success at all evaluated team sizes, but remains below the proposed method under the matched training horizon and final-window evaluation protocol.

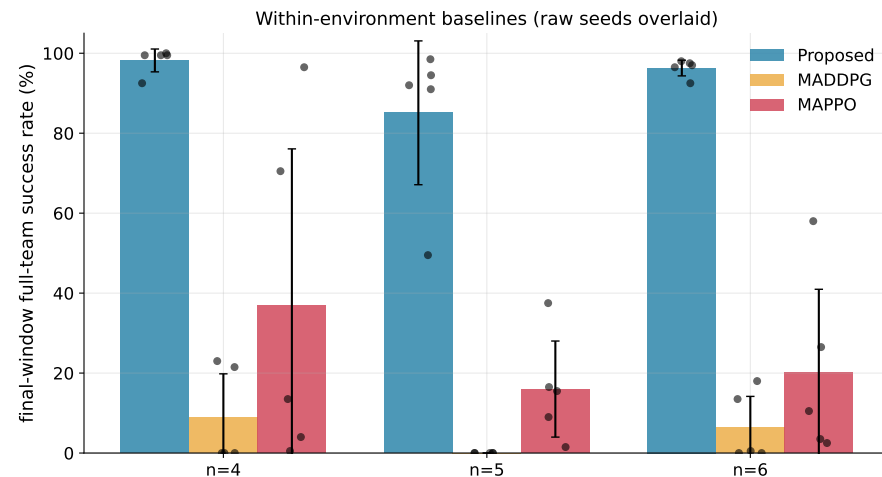


Figure 22. Within-environment baseline comparison between the proposed full configuration, MADDPG, and MAPPO. Bars show mean final-window full-team success over five independent seeds; error bars denote standard deviation across seeds; overlaid points show individual seed-level success rates.

For additional context, and separately from the peer-baseline comparison in Figure 22, we also evaluated a centralized Hungarian-assignment proportional controller as an oracle upper bound. This controller receives global assignment information and full-state access at each step, so it is not treated as a fair decentralized peer baseline. In the obstacle-free training arena, the oracle reached 100% success and 100% coverage for $n = 4$ –6 with mean completion around 10 steps.

3.14. Environment-Size Transfer and Computational Cost

To evaluate transfer across arena sizes, policies trained in the 20×20 arena were evaluated without retraining using deterministic rollouts in arenas of size 15×15 , 18×18 , 20×20 , 25×25 , and 30×30 . This sweep covers smaller arenas, the training arena, and larger arenas up to a 50% increase in linear arena size. Table 9 reports full-team success across the arena-size sweep, while Figure 23 summarizes both full-team success and partial landmark coverage.

Table 9. Environment-size transfer for policies trained in the 20×20 arena and evaluated without retraining. Values report deterministic full-team success rate as mean $\pm$ standard deviation across five seeds.

n	env15 SR (%)	env18 SR (%)	env20 SR (%)	env25 SR (%)	env30 SR (%)
4	98.4 $\pm$ 1.0	94.9 $\pm$ 5.5	92.8 $\pm$ 6.0	84.1 $\pm$ 11.8	70.7 $\pm$ 9.3
5	83.5 $\pm$ 27.5	75.3 $\pm$ 33.9	69.9 $\pm$ 32.4	55.7 $\pm$ 27.9	41.2 $\pm$ 22.5
6	94.0 $\pm$ 3.6	91.1 $\pm$ 6.1	89.9 $\pm$ 6.9	71.4 $\pm$ 8.5	48.2 $\pm$ 9.7

As shown in Table 9 and Figure 23, full-team success decreases as the arena becomes larger than the training arena. The degradation is gradual rather than abrupt over the tested range, but it is more pronounced for $n = 5$ and $n = 6$ at 30×30 . Coverage remains higher than full-team success across the sweep, indicating that the learned policies often

reach most landmarks even when full-team completion is not achieved within the horizon. These results provide a deterministic arena-size transfer check over the tested range only and should not be interpreted as broad environment generalization.

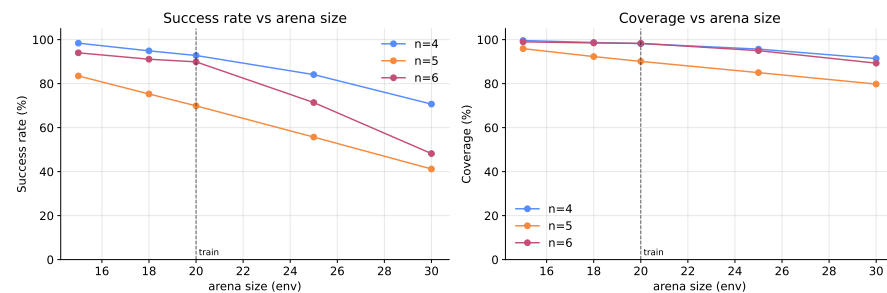


Figure 23. Deterministic environment-size transfer evaluation for policies trained in the 20×20 arena. Curves show full-team success rate and landmark coverage across arena sizes without retraining.

Table 10 summarizes the computational cost of the proposed-method training sweep.

Table 10. Computational cost, model size, and reproducibility summary for the proposed-method training sweep.

Item	Value
GPU	NVIDIA RTX 4050 Laptop GPU (NVIDIA Corporation, Santa Clara, CA, USA), 6 GB
Software	Python 3.12.3; CUDA 12.0; NVIDIA driver
Proposed-method training sweep	610.43.02
Independent seeds	45 runs
Episodes per run	5
Mean training time	1000
Total training compute	4.9 GPU-h/run
Peak GPU memory	≈ 221 GPU-h
Actor parameters	≤ 0.074 GB/run
Critic parameters	73,986 ($n = 4$); 76,034 ($n = 5$); 78,082 ($n = 6$)
Hungarian assignment cost	20,737 ($n = 4$); 21,761 ($n = 5$); 22,785 ($n = 6$)
	$O(n^3)$ per environment step

The Hungarian assignment is inexpensive for the evaluated team sizes $n \leq 6$, but its $O(n^3)$ per-step cost may become non-negligible for substantially larger teams.

4. Discussion

The results show that the proposed shared actor–critic framework can solve cooperative coverage tasks with small homogeneous teams in the evaluated obstacle-free simulation. In the five-seed final-window evaluation, the full reward configuration achieved full-team success rates of $98.2 \pm 2.9\%$, $85.1 \pm 18.0\%$, and $96.3 \pm 2.0\%$ for $n = 4, 5, 6$, respectively, with mean coverage above 96% in all three cases. These results show both high average performance and non-negligible seed sensitivity, particularly for full-team success. The reward ablations also require a cautious interpretation: the no-assignment-shaping and no-collision variants remained viable, and the seed-level tests in Section 3.10 did not show a statistically significant final-window advantage of the full reward configuration over the ablated configurations. The main measured effect of assignment-aware shaping is instead sample efficiency: as reported in Section 3.6, the full configuration reached the 80% rolling-success threshold earlier in median terms, with the clearest seed-level statistical support at $n = 4$.

A central observation is that training leads to qualitatively distinct coordination strategies in the evaluated setting. Early-phase trajectories were characterized by oscillatory motion, frequent heading reversals, and extended exploration, which corresponded to large penalties in time, heading misalignment, and collisions. By contrast, late-phase trajectories revealed decisive motion patterns in which agents rapidly committed to assignments and advanced directly to their designated landmarks with limited corrective turning. This behavioral transition was quantitatively corroborated by lower late-phase Hungarian distances, the emergence of burst-settle velocity profiles, and the reduction of collision-penalty magnitude during training. These trends indicate that the shared actor-critic architecture can learn policies that reduce unnecessary motion patterns and improve task completion in the tested setting. The path-length values in Section 3.9 provide additional quantitative context for the trajectory examples, while the absence of external static obstacles and hardware trials means that the findings should be interpreted as collision-aware behavior in simulation rather than as deployment-level safety evidence.

The statistical analysis is based on independent training seeds rather than treating episodes from a single training trajectory as independent samples. Seed-level Kruskal–Wallis tests did not show significant differences among the full, no-assignment-shaping, and no-collision configurations for any evaluated team size, and pooled Mann–Whitney tests did not show a significant advantage of the full configuration over either ablation. Because the five-seed tests have low power, these non-significant results are not interpreted as evidence of equivalence or as evidence that the ablated components are unnecessary. This result limits the final-window ablation claim: the proposed reward design is useful as a shaping mechanism, but the present evidence does not establish that every shaping component is necessary for fixed-budget final-window full-team success. Completion-time behavior is reported using deterministic empirical completion-time curves in the 20×20 training arena, while arena-size transfer is analyzed separately in Section 3.14.

The reward-weight choices should also be interpreted as empirical design choices. The weights in Table 1 were selected during preliminary tuning and then kept fixed across the reported team sizes and seeds. The collision-kernel sharpness parameter $c = 7$ was likewise treated as a fixed hyperparameter. The sensitivity analysis in Section 3.12 shows that the converged score is nearly unchanged around this value for $n = 4$ and $n = 5$, while the $n = 6, c = 15$ setting is less stable. Thus, the reported value is supported as a stable empirical setting for these experiments, but it is not claimed to be theoretically optimal, and the reported sweep should not be interpreted as a full reward-weight sensitivity study.

The team-size results should be interpreted only over the evaluated range $n \in \{4, 5, 6\}$. The experiments do not establish swarm-scale scalability, because larger teams such as $n = 8$ or $n = 10$ were not evaluated. Within the tested range, the shared-parameter design supports a common policy and value-function parameterization for homogeneous agents and allows experience to be pooled across agents. The implementation uses a single shared critic applied to local observation–action pairs; it is therefore not a centralized CTDE critic and does not explicitly estimate joint state–action values or solve multi-agent credit assignment. The Hungarian assignment calculation is also recomputed at each environment step and has $O(n^3)$ complexity. This cost is small for $n \leq 6$, but it may become relevant for substantially larger teams. Therefore, the present evidence supports performance over the tested small-team setting, while larger-team scaling remains future work.

The lower mean success at $n = 5$ is primarily explained by seed-level variability rather than by a consistent degradation across all $n = 5$ runs. The per-seed diagnostic in Section 3.8 shows that one low-success seed dominates the lower aggregate mean, while all formal summaries retain all five seeds. This pattern suggests a seed-sensitive orbiting or stalling failure mode rather than a demonstrated structural odd-team-size limitation. The training-

pipeline ablation in Section 3.7 is consistent with this interpretation, because removing restart handling produced seed-specific collapses rather than uniform degradation across all seeds.

The contextual comparisons with CRAMP and MASP are retained only to position the task relative to related grid-based and particle-based benchmarks. They should not be read as head-to-head evidence because the environments, dynamics, horizons, observation models, and success definitions differ. The direct baseline evidence in this study is the within-environment comparison with MADDPG and MAPPO reported in Section 3.13. Under the tested protocol, MADDPG and MAPPO achieved substantially lower full-team success than the proposed shared actor–critic configuration. MAPPO improved over MADDPG but remained below the proposed method for all evaluated team sizes. These comparisons remain limited to the same simplified obstacle-free coverage environment.

The qualitative analyses provide diagnostic evidence about the learned behavior. Agents tend to exhibit rapid commitment to assignments followed by more stable approach trajectories, which is consistent with the reduction in distance-to-assignment and collision-penalty activity observed during training. The reported path-length metric gives a compact numerical complement to these trajectory plots. However, these qualitative results are examples from simulation and do not establish behavior in cluttered, dynamic, physical, or aerial-robot settings. Extending the framework to external obstacles, richer sensing, 3D motion, and hardware validation remains necessary before making deployment-level claims.

Taken together, the results support a narrow empirical claim: the proposed integrated training pipeline is effective for the evaluated obstacle-free small-team coverage tasks, but its generalization beyond this setting remains untested. The conclusions should therefore be limited to the evaluated simulation protocol, with broader validation left for future work.

5. Conclusions

This paper presented a shared actor–critic reinforcement learning framework for decentralized coordination in small homogeneous multi-robot coverage tasks. The method combines local observations, continuous differential-drive control, shared policy/value parameters, and assignment-aware reward shaping within an obstacle-free simulated coverage environment.

Across the evaluated team sizes $n \in \{4, 5, 6\}$, the full reward configuration produced high final-window full-team success and consistently high landmark coverage. The $n = 5$ setting was the least stable case, with the lower mean primarily reflecting seed-level variability rather than uniformly poor performance across all runs. These results support the effectiveness of the proposed method in the tested small-team setting, while keeping the conclusion limited to the evaluated range.

The ablation results show that the shaping terms should be interpreted cautiously. The ablated configurations remained viable in final-window evaluation, and the seed-level tests did not establish a significant final-window advantage of the full reward configuration. The clearest measured contribution of assignment-aware shaping was instead in learning speed, where the full configuration reached the 80% rolling-success threshold earlier in median terms, with the strongest seed-level support at $n = 4$. The HER/restart ablation further indicates that the reported performance reflects an integrated training pipeline rather than a reward-only effect.

Within-environment comparisons showed that the proposed configuration outperformed MADDPG and MAPPO under the matched training budget and final-window protocol. Deterministic arena-size transfer over 15×15 to 30×30 arenas showed decreasing

full-team success as arena size increased, while partial coverage remained higher than strict full-team completion. These findings indicate useful transfer over the tested range, but not broad environment generalization.

The scope of these conclusions remains specific to obstacle-free simulation, small team sizes, and abstract differential-drive dynamics. The results should not be interpreted as demonstrating formal convergence, deployment-level safety, platform-agnostic operation, or swarm-scale scalability. Future work should extend the evaluation to larger teams, obstacle-rich environments, broader MARL and MAPF-related baselines, UAV-specific dynamics, and hardware or high-fidelity robotic validation.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/robotics15070119/s1>.

Author Contributions: Conceptualization, A.E.K., D.Y. and S.A.; methodology, A.E.K.; software, A.E.K.; validation, A.E.K., D.Y. and S.N.; formal analysis, A.E.K. and S.A.; investigation, A.E.K.; resources, D.Y. and S.N.; data curation, A.E.K.; writing—original draft preparation, A.E.K.; writing—review and editing, A.E.K., D.Y., S.A. and S.N.; visualization, A.E.K.; funding acquisition, S.N. and S.A. All authors have read and agreed to the published version of this manuscript.

Funding: This research was funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan: Grant AP23488281, “Development of an automatic drone charging system based on Reinforcement Learning technology, which allows for continuous monitoring of vast territories”; Grant BR24992852, “Intelligent models and methods of Smart City digital ecosystem for sustainable development and the citizens’ quality of life improvement”.

Data Availability Statement: The multi-seed training logs, selected baseline logs, and deterministic evaluation summaries are provided as Supplementary Materials. Additional data related to this work are available from the authors upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used OpenAI ChatGPT (GPT-5.5 Thinking) for language polishing, grammar correction, sentence-level clarity, punctuation, and formatting assistance of author-drafted text. The tool was not used to generate experimental data, source code, figures, study design, data analysis, data interpretation, scientific claims, or conclusions. All scientific content, methodological design, experimental settings, analyses, results, interpretations, and conclusions were developed, reviewed, revised, and validated by the authors, who take full responsibility for the final content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
COMA	Counterfactual Multi-Agent Policy Gradients
CR	Collision Rate
CRAMP	Communication-Reduced Attention-based Multi-Agent Pathfinding
CTDE	Centralized Training with Decentralized Execution
DDPG	Deep Deterministic Policy Gradient
DRL	Deep Reinforcement Learning
HER	Hindsight Experience Replay
IPPO	Independent Proximal Policy Optimization
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MAPF	Multi-Agent Path Finding
MAPPO	Multi-Agent Proximal Policy Optimization
MARL	Multi-Agent Reinforcement Learning
MASP	Multi-Agent Spatial Planning

MDP	Markov Decision Process
MPE	Multi-Agent Particle Environment
MRTA	Multi-Robot Task Allocation
PPO	Proximal Policy Optimization
QMIX	Monotonic Value Factorization for Deep Multi-Agent Reinforcement Learning
RL	Reinforcement Learning
SAC	Soft Actor–Critic
SMAC	StarCraft Multi-Agent Challenge
SR	Success Rate
TD3	Twin Delayed Deep Deterministic Policy Gradient
UAV	Unmanned Aerial Vehicle
VDN	Value Decomposition Networks
VMAS	Vectorized Multi-Agent Simulator

References

1. van Otterlo, M.; Wiering, M.A. Reinforcement Learning and Markov Decision Processes. In *The Logic of Adaptive Behavior*; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2012. [\[CrossRef\]](#)
2. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
3. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
4. Tang, C.; Abbatematteo, B.; Hu, J.; Chandra, R.; Martín-Martín, R.; Stone, P. Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes. *Annu. Rev. Control. Robot. Auton. Syst.* **2025**, *8*, 153–188. [\[CrossRef\]](#)
5. Singh, B.; Kapoor, R.; Kar, I. Reinforcement Learning in Robotic Applications. *Artif. Intell. Rev.* **2022**, *55*, 4757–4807. [\[CrossRef\]](#)
6. Han, D.; Hu, W.; Ma, H.; Fang, H.; Liu, Y.; Chen, T. A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation. *Sensors* **2023**, *23*, 3762. [\[CrossRef\]](#) [\[PubMed\]](#)
7. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. *arXiv* **2018**, arXiv:1812.11794.
8. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943. [\[CrossRef\]](#)
9. Huh, D.; Mohapatra, P. Multi-Agent Reinforcement Learning: A Comprehensive Survey. *arXiv* **2023**, arXiv:2312.10256.
10. Zhu, C.; Dastani, M.; Wang, S. A survey of multi-agent deep reinforcement learning with communication. *Auton. Agents Multi-Agent Syst.* **2024**, *38*, 4. [\[CrossRef\]](#)
11. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 750–797. [\[CrossRef\]](#)
12. Zhang, K.; Yang, Z.; Başar, T. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In *Handbook of Reinforcement Learning and Control*; Studies in Systems, Decision and Control; Springer: Cham, Switzerland, 2021; Volume 325, pp. 321–384. [\[CrossRef\]](#)
13. Hua, M.; Chen, D.; Qi, X.; Jiang, K.; Liu, Z.E.; Zhou, Q.; Xu, H. Multi-Agent Reinforcement Learning for Connected and Automated Vehicles Control: Recent Advancements and Future Prospects. *arXiv* **2023**, arXiv:2312.11084.
14. Orr, J.; Dutta, A. Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey. *Sensors* **2023**, *23*, 3625. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Debie, E.; Dorri, A.; Kanhere, S.S.; Jurdak, R. Swarm Robotics: A Survey from a Multi-Tasking Perspective. *ACM Comput. Surv.* **2023**, *56*, 49. [\[CrossRef\]](#)
16. Blais, M.A.; Akhloufi, M.A. Reinforcement learning for swarm robotics: An overview of applications, algorithms and simulators. *Cogn. Robot.* **2023**, *3*, 226–256. [\[CrossRef\]](#)
17. Seitbattalov, Z.Y.; Canbolat, H.; Moldabayeva, Z.S.; Kyzyrkanov, A.E. An Intelligent Automatic Number Plate Recognition System Based on Computer Vision and Edge Computing. In Proceedings of the 2022 International Conference on Smart Information Systems and Technologies (SIST), Nur-Sultan, Kazakhstan, 28–30 April 2022; pp. 1–5. [\[CrossRef\]](#)
18. Toibazarov, D.; Baiseitov, G.; Kyzyrkanov, A.; Aljawarneh, S.; Atanov, S. Development of control software for self-organizing intelligent mobile robots. *East.-Eur. J. Enterp. Technol.* **2023**, *2*, 46–58. [\[CrossRef\]](#)
19. Kyzyrkanov, A.; Atanov, S.; Aljawarneh, S. Coordination of movement of multiagent robotic systems. In Proceedings of the 2021 16th International Conference on Electronics Computer and Computation (ICECCO), Kaskelen, Kazakhstan, 25–26 November 2021; pp. 1–4. [\[CrossRef\]](#)
20. Kyzyrkanov, A.E.; Atanov, S.K.; Aljawarneh, S.A.R. Formation control and coordination of swarm robotic systems. In Proceedings of the 7th International Conference on Engineering & MIS 2021, Almaty, Kazakhstan, 11–13 October 2021; pp. 1–11. [\[CrossRef\]](#)

21. Kyzyrkanov, A.E.; Atanov, S.K.; Aljawarneh, S.A.; Tursynova, N.A. Movement Coordination of Swarm Robotic Systems. In Proceedings of the 7th International Conference on Digital Technologies in Education, Science and Industry (DTESI 2022), Almaty, Kazakhstan, 20–21 October 2022; Volume 3382.
22. Kyzyrkanov, A.; Atanov, S.; Aljawarneh, S.; Tursynova, N.; Kassymkhanov, S. Algorithm of Coordination of Swarm of Autonomous Robots. In Proceedings of the 2023 IEEE International Conference on Smart Information Systems and Technologies (SIST), Astana, Kazakhstan, 4–6 May 2023; pp. 539–544. [\[CrossRef\]](#)
23. Kyzyrkanov, A.E.; Atanov, S.K.; Aljawarneh, S.A.; Tursynova, N.A. Intelligent Control of a Swarm of Reconnaissance Robots for Terrain Monitoring Tasks. In Proceedings of the Workshops and Short Paper Proceedings of the 8th International Conference on Digital Technologies in Education, Science and Industry 2023 (DTESI 2023), Almaty, Kazakhstan, 6–7 December 2024; Volume 3680.
24. Kyzyrkanov, A.E.; Tursynova, N.A.; Yedilkhan, D.; Otarbay, Z.; Atanov, S.; Aljawarneh, S. Intelligent Coordination for a Swarm of Autonomous Mobile Robots. *Procedia Comput. Sci.* **2024**, *241*, 464–469. [\[CrossRef\]](#)
25. Tang, R.; Tang, J.; Abu Talip, M.S.; Aridas, N.K.; Xu, X. Enhanced Multi-Agent Coordination Algorithm for Drone Swarm Patrolling in Durian Orchards. *Sci. Rep.* **2025**, *15*, 9139. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Qin, C.; Pournaras, E. Coordination of Drones at Scale: Decentralized Energy-aware Swarm Intelligence for Spatio-temporal Sensing. *Transp. Res. Part C Emerg. Technol.* **2023**, *157*, 104387. [\[CrossRef\]](#)
27. Xia, B.; Mantegh, I.; Xie, W. Decentralized UAV Swarm Control: A Multi-Layered Architecture for Integrated Flight Mode Management and Dynamic Target Interception. *Drones* **2024**, *8*, 350. [\[CrossRef\]](#)
28. Alqudsi, Y.; Makaraci, M. UAV swarms: Research, challenges, and future directions. *J. Eng. Appl. Sci.* **2025**, *72*, 12. [\[CrossRef\]](#)
29. Yedilkhan, D.; Kyzyrkanov, A.; Amirgaliyev, B.; Khaimuldin, N.; Ayub, M.S.; Zhumadillayeva, A. Efficient Area Coverage Strategies for High-Altitude UAVs in Smart City Monitoring. *Drones* **2025**, *9*, 632. [\[CrossRef\]](#)
30. Li, J.; Li, J.; Zhang, J.; Meng, W. A Comprehensive Review of Path-Planning Algorithms for Multi-UAV Swarms. *Drones* **2026**, *10*, 11. [\[CrossRef\]](#)
31. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
32. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-Decomposition Networks For Cooperative Multi-Agent Learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018*; International Foundation for Autonomous Agents and Multiagent Systems: Pittsburgh, PA, USA, 2018; pp. 2085–2087.
33. Rashid, T.; Samvelyan, M.; de Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018*; Proceedings of Machine Learning Research; PMLR: Cambridge, MA, USA, 2018; Volume 80, pp. 4295–4304.
34. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual Multi-Agent Policy Gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
35. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347. [\[CrossRef\]](#)
36. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. *arXiv* **2021**, arXiv:2103.01955. [\[CrossRef\]](#)
37. de Witt, C.S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.S.; Sun, M.; Whiteson, S. Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge? *arXiv* **2020**, arXiv:2011.09533. [\[CrossRef\]](#)
38. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018*; Proceedings of Machine Learning Research; PMLR: Cambridge, MA, USA, 2018; Volume 80, pp. 1587–1596.
39. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018*; Proceedings of Machine Learning Research; PMLR: Cambridge, MA, USA, 2018; Volume 80, pp. 1861–1870.
40. Samvelyan, M.; Rashid, T.; de Witt, C.S.; Farquhar, G.; Nardelli, N.; Rudner, T.G.J.; Hung, C.M.; Torr, P.H.S.; Foerster, J.; Whiteson, S. The StarCraft Multi-Agent Challenge. *arXiv* **2019**, arXiv:1902.04043. [\[CrossRef\]](#)
41. Ellis, B.; Cook, J.; Moalla, S.; Samvelyan, M.; Sun, M.; Mahajan, A.; Foerster, J.N.; Whiteson, S. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 37567–37593. [\[CrossRef\]](#)
42. Bettini, M.; Kortvelesy, R.; Blumenkamp, J.; Prorok, A. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. In Proceedings of the 16th International Symposium on Distributed Autonomous Robotic Systems, Montbéliard, France, 28–30 November 2022. [\[CrossRef\]](#)

43. Bagchi, M.J.; Nair, S.B.; Das, P.K. On a Dynamic and Decentralized Energy-aware Technique for Multi-robot Task Allocation. *Robot. Auton. Syst.* **2024**, *180*, 104762. [\[CrossRef\]](#)
44. Shan, X.; Jin, Y.; Jurt, M.; Hirche, S. A Distributed Multi-robot Task Allocation Method for Time-constrained Dynamic Collective Transport. *Robot. Auton. Syst.* **2024**, *178*, 104722. [\[CrossRef\]](#)
45. Alemu, E.; Wang, W.; Shi, G.; Wang, J.; Li, B.; Yuan, Z. A Systematic Literature Review on Multi-Robot Task Allocation. *ACM Comput. Surv.* **2024**, *57*, 68. [\[CrossRef\]](#)
46. Chakraa, H.; Guérin, F.; Leclercq, E.; Lefebvre, D. Optimization Techniques for Multi-Robot Task Allocation Problems: Review on the State-of-the-Art. *Robot. Auton. Syst.* **2023**, *168*, 104492. [\[CrossRef\]](#)
47. Aziz, H.; Chan, H.; Cseh, Á.; Li, B.; Ramezani, F.; Wang, C. Multi-Robot Task Allocation – Complexity and Approximation. *arXiv* **2021**, arXiv:2103.12370. [\[CrossRef\]](#)
48. Shin, H.S.; Li, T.; Segui-Gasco, P. Sample Greedy Based Task Allocation for Multiple Robot Systems. *arXiv* **2019**, arXiv:1901.03258. [\[CrossRef\]](#)
49. Huang, S.; Zhang, H.; Huang, Z. E2CoPre: Energy Efficient and Cooperative Collision Avoidance for UAV Swarms with Trajectory Prediction. *arXiv* **2023**, arXiv:2303.06510. [\[CrossRef\]](#)
50. Bu, Y.; Xiao, D.; Liu, Y.; Zhou, M.; Wu, Z.; Li, H.; Zhang, K. Advancement Challenges in UAV Swarm Formation Control: A Comprehensive Review. *Drones* **2024**, *8*, 320. [\[CrossRef\]](#)
51. Marek, D.; Biernacki, P.; Szygła, J.; Domański, A.; Wojciechowski, K. Collision Avoidance Mechanism for Swarms of Drones. *Sensors* **2025**, *25*, 1141. [\[CrossRef\]](#) [\[PubMed\]](#)
52. Huang, S.; Zhang, H.; Huang, Z. Multi-UAV Collision Avoidance using Multi-Agent Reinforcement Learning with Counterfactual Credit Assignment. *arXiv* **2022**, arXiv:2204.08594.
53. Muhsen, D.K.; Al-Azawi, A.F.; Hashim, S.H.; Jebur, M.A. A Survey on Swarm Robotics for Area Coverage Problem. *Algorithms* **2023**, *17*, 3. [\[CrossRef\]](#)
54. Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.K.S.; et al. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *arXiv* **2019**, arXiv:1906.08291. [\[CrossRef\]](#)
55. Lam, E.; Le Bodic, P.; Stuckey, P.J.; Harabor, D.D.; Gange, G. Branch-and-Cut-and-Price for Multi-Agent Path Finding. *Comput. Oper. Res.* **2022**, *144*, 105809. [\[CrossRef\]](#)
56. Achá, D.; Grastien, A.; Stuckey, P.J.; Guns, T. Multi-Agent Path Finding: A New Boolean Encoding. *J. Artif. Intell. Res.* **2022**, *75*, 323–350. [\[CrossRef\]](#)
57. Zhang, J.; Li, J.; Botea, A.; Li, J.; Harabor, D.; Stuckey, P. Efficient Multi-Agent Path Finding with Turn Actions. In Proceedings of the 16th Annual Symposium on Combinatorial Search (SOCS-23), Prague, Czech Republic, 14–16 July 2023.
58. Makino, H.; Ohama, Y.; Ito, S. MARPF: Multi-Agent and Multi-Rack Path Finding. In Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates, 14–18 October 2024. [\[CrossRef\]](#)
59. Pham, P.; Bera, A. Optimizing Crowd-Aware Multi-Agent Path Finding through Local Communication with Graph Neural Networks. *arXiv* **2023**, arXiv:2309.10275. [\[CrossRef\]](#)
60. Yakovlev, K.S.; Andreychuk, A. Planning and Learning in Multi-Agent Path Finding. *Dokl. Math.* **2022**, *106*, 574–578. [\[CrossRef\]](#)
61. Liang, J.; Koenig, S.; Ma, H.; Atzmon, Y.; Ren, Z. Multi-Agent Path Finding in Continuous Spaces with Projected Diffusion Models. *arXiv* **2024**, arXiv:2412.17993.
62. Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T.K.S.; Koenig, S.; Choset, H. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *arXiv* **2018**, arXiv:1809.03531. [\[CrossRef\]](#)
63. Wang, Y.; Xiang, B.; Huang, S.; Sartoretti, G. SCRIMP: Scalable Communication for Reinforcement- and Imitation-Learning-Based Multi-Agent Pathfinding. *arXiv* **2023**, arXiv:2303.00605. [\[CrossRef\]](#)
64. Ni, J.; Gu, Y.; Tang, G.; Ke, C.; Gu, Y. Cooperative Coverage Path Planning for Multi-Mobile Robots Based on Improved K-Means Clustering and Deep Reinforcement Learning. *Electronics* **2024**, *13*, 944. [\[CrossRef\]](#)
65. Chen, Y.; Lu, Z.M.; Cui, J.L.; Luo, H.; Zheng, Y.M. A Complete Coverage Path Planning Algorithm for Lawn Mowing Robots Based on Deep Reinforcement Learning. *Sensors* **2025**, *25*, 416. [\[CrossRef\]](#) [\[PubMed\]](#)
66. Ali, M.A.; Maqsood, A.; Athar, U.; Ali, S. Benchmarking control strategies for UAV swarms: Centralized, decentralized, or federated reinforcement learning. *Aerosp. Sci. Technol.* **2026**, *170*, 111539. [\[CrossRef\]](#)
67. Mead, T.; Wang, Z.; Foo, E.; Dong, J.S.; Dong, N.; Ko, R.; Koay, A.M.; Nguyen, K.; Xu, Y.; Kim, J.; et al. Multi-agent reinforcement curriculum learning for real unmanned ground vehicles. *Eng. Appl. Artif. Intell.* **2026**, *167*, 113780. [\[CrossRef\]](#)
68. Nurakhov, Y.; Kyzyrkanov, A.; Otarbay, Z.; Lebedev, D. Adaptive Software-Defined Network Control Using Kernel-Based Reinforcement Learning: An Empirical Study. *Appl. Sci.* **2025**, *15*, 12349. [\[CrossRef\]](#)
69. Kuhn, H.W. The Hungarian Method for the Assignment Problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [\[CrossRef\]](#)
70. Munkres, J. Algorithms for the Assignment and Transportation Problems. *J. Soc. Ind. Appl. Math.* **1957**, *5*, 32–38. [\[CrossRef\]](#)

71. Zhang, S.; Xue, Y.; Zhang, H.; Zhou, X.; Li, K.; Liu, R. Improved Hungarian Algorithm-Based Task Scheduling Optimization Strategy for Remote Sensing Big Data Processing. *Geo-Spat. Inf. Sci.* **2024**, *27*, 1141–1154. [[CrossRef](#)]
72. Lindsay, N.; Buehling, R.K.; Sun, L. A Sequential Task Addition Distributed Assignment Algorithm for Multi-Robot Systems. *J. Intell. Robot. Syst.* **2021**, *102*, 51. [[CrossRef](#)]
73. Goarin, M.; Loianno, G. Graph Neural Network for Decentralized Multi-Robot Goal Assignment. *IEEE Robot. Autom. Lett.* **2024**, *9*, 4051–4058. [[CrossRef](#)]
74. Kuba, J.G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; Yang, Y. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. *arXiv* **2021**, arXiv:2109.11251.
75. Liu, X.; Li, X.; Li, Y.; Huang, Y.; Wang, Z. PS-QMix: A Parallel Learning Framework for Q-Mix Using Parameter Server. In *Proceedings of the Advanced Data Mining and Applications (PAKDD 2022)*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2022; pp. 341–352. [[CrossRef](#)]
76. Zhao, T.; Chen, T.; Zhang, B. QMIX-GNN: A Graph Neural Network-Based Heterogeneous Multi-Agent Reinforcement Learning Model for Improved Collaboration and Decision-Making. *Appl. Sci.* **2025**, *15*, 3794. [[CrossRef](#)]
77. Wei, X.; Cui, W.; Huang, X.; Yang, L.; Tao, Z.; Wang, B. Graph MADDPG with RNN for Multiagent Cooperative Environment. *Front. Neurobot.* **2023**, *17*, 1185169. [[CrossRef](#)] [[PubMed](#)]
78. Yang, X.; Yang, X.; Yu, C.; Chen, J.; Ding, W.; Yang, H.; Wang, Y. MASP: Scalable GNN-based Planning for Multi-Agent Navigation. *arXiv* **2024**, arXiv:2312.02522. [[CrossRef](#)]
79. Buşoniu, L.; Babuška, R.; De Schutter, B. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2008**, *38*, 156–172. [[CrossRef](#)]
80. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Abbeel, P.; Zaremba, W. Hindsight Experience Replay. In *Proceedings of the Advances in Neural Information Processing Systems*, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
81. Kruskal, W.H.; Wallis, W.A. Use of Ranks in One-Criterion Variance Analysis. *J. Am. Stat. Assoc.* **1952**, *47*, 583–621. [[CrossRef](#)]
82. Mann, H.B.; Whitney, D.R. On a Test of Whether One of Two Random Variables Is Stochastically Larger than the Other. *Ann. Math. Stat.* **1947**, *18*, 50–60. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.