AC-MASAC: An Attentive Curriculum Learning Framework for Heterogeneous UAV Swarm Coordination

Wanhao Liu
Guangdong University of Technology
Guangzhou, China
3123006457@mail2.gdut.edu.cn

Junhong Dai
Guangdong University of Technology
Guangzhou, China
2112304096@gdut.edu.cn

Yixuan Zhang
Guangdong University of Technology
Guangzhou, China
zhangyixuan58@mails.gdut.edu.cn

Shengyun Yin
Guangdong University of Technology
Guangzhou, China
2832003829@mails.gdut.edu.cn

Panshuo Li*
Guangdong University of Technology
Guangzhou, China
panshuoli812@gdut.edu.cn

ABSTRACT

Cooperative path planning for heterogeneous UAV swarms poses significant challenges for Multi-Agent Reinforcement Learning (MARL), particularly in handling asymmetric inter-agent dependencies and addressing the risks of sparse rewards and catastrophic forgetting during training. To address these issues, this paper proposes an attentive curriculum learning framework (AC-MASAC). The framework introduces a role-aware heterogeneous attention mechanism to explicitly model asymmetric dependencies. Moreover, a structured curriculum strategy is designed, integrating hierarchical knowledge transfer and stage-proportional experience replay to address the issues of sparse rewards and catastrophic forgetting. The proposed framework is validated on a custom multi-agent simulation platform, and the results show that our method has significant advantages over other advanced methods in terms of Success Rate, Formation Keeping Rate, and Success-weighted Mission Time. The code is available at https://github.com/Wanhao-Liu/AC-MASAC and the extended version with the appendix can be found at https://arxiv.org/abs/2602.11735.

KEYWORDS

Heterogeneous multi-agent systems; Reinforcement learning; UAV path planning; Attention mechanisms; Curriculum learning

ACM Reference Format:
Wanhao Liu, Junhong Dai, Yixuan Zhang, Shengyun Yin, and Panshuo Li. 2026. AC-MASAC: An Attentive Curriculum Learning Framework for Heterogeneous UAV Swarm Coordination. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 8 pages. https://doi.org/10.65109/VJGN3439

*Corresponding author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/VJGN3439

1 INTRODUCTION

Unmanned Aerial Vehicle (UAV) swarms play an increasingly vital role in coordinated tasks such as logistics, inspection, and search-and-rescue [8]. Central to these applications is the challenge of cooperative path planning—generating collision-free, dynamically feasible trajectories for all agents to achieve a common goal, as illustrated in Figure 1. Although this is a widely researched area with numerous existing solutions [17, 22, 24], current methods still face limitations when addressing specific challenges.

Historically, approaches to this problem can be broadly divided into traditional and learning-based methods. Traditional methods, largely derived from robotics research, include techniques such as sampling-based planners (e.g., RRT* [11, 13]) and potential field methods (e.g., APF [12]). While valuable in certain contexts, these approaches often share common limitations: they typically require a precise model of the environment, exhibit limited adaptability to unforeseen dynamic events, and face significant computational scaling challenges as the swarm size grows.

To overcome the challenges of model dependency and poor adaptability, Multi-Agent Reinforcement Learning (MARL) has emerged as a powerful paradigm for coordinating UAV swarms. Early approaches often treated agents as independent learners (ILs), where each agent learned its policy using standard single-agent RL algorithms like Q-learning or DDPG, considering other agents as part of a dynamic environment [21]. However, this method struggles with the non-stationarity problem, as the concurrent evolution of all agents' policies makes the environment highly unstable and hinders convergence. To address this core challenge, the Centralized Training with Decentralized Execution (CTDE) framework has become the dominant paradigm in the field [5, 14]. Within this framework, one prominent line of research focuses on value-decomposition methods. These approaches learn a centralized but factorizable value function, starting with foundational techniques like Value-Decomposition Networks (VDN) [20] and evolving to more expressive methods such as QMIX [16] and QTRAN [18]. While powerful, these methods are primarily designed for discrete action spaces because selecting an optimal action requires finding the argmax over the joint action space, which becomes an intractable optimization problem in continuous domains. Consequently, Actor-Critic based methods have proven more effective for tasks requiring continuous control. They decouple the problem by learning a dedicated policy network (the actor) that directly maps

states to continuous actions, while the critic evaluates the actions provided by the actors. This principle was successfully extended to the multi-agent domain by the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm [14], which builds upon DDPG—a method inherently designed for continuous control. Following this line of work, and to address the overestimation bias often found in single-critic methods, the MATD3 algorithm was proposed in [1]. It extends the clipped double-Q learning and delayed policy updates from the single-agent TD3 algorithm to the multi-agent setting, leading to more stable value estimates. These foundational Actor-Critic approaches established a powerful and flexible framework for tasks involving continuous control in MARL.

While the above mentioned methods were effective, they often faced challenges with training stability and sample efficiency. To address these issues, the use of Soft Actor-Critic (SAC) [6] has been explored in the multi-agent context. The MASAC algorithm, as presented in [4, 7], incorporates the maximum entropy objective to encourage more thorough exploration, leading to more robust and efficient coordination policies. Furthermore, as the research focus shifted towards larger-scale swarms, the challenge of effectively processing and filtering the vast amount of peer information became prominent. To this end, attention mechanisms were integrated into the framework. An attention-based critic was introduced in [9], allowing agents to selectively focus on more relevant neighbors when evaluating state-values. Going a step further, attention was applied to the actor network in [19], enabling agents to learn a communication strategy to actively select which peers to broadcast their information to. These methods, within a typical homogeneous swarm setting, have significantly improved decision-making in dense and complex scenarios.

Despite these significant advances, applying these frameworks to more complex, structured swarm scenarios reveals two key open challenges. On the one hand, much of the foundational work has centered on homogeneous agent systems, where agents are considered interchangeable. This approach, while broadly applicable, presents limitations when modeling heterogeneous swarms (e.g., Leader-Follower teams), which are characterized by asymmetric inter-agent dependencies that are not explicitly captured by standard models. On the other hand, the risks associated with sparse rewards and catastrophic forgetting remain primary obstacles to achieving stable and efficient policy convergence when applying these algorithms directly to complex tasks. To address these challenges of heterogeneity and training stability, this paper presents the AC-MASAC algorithm. The contributions of this work are summarized as follows:

- We propose a heterogeneous attention model within the MASAC framework that explicitly models the Leader-Follower dynamic. Its role-aware information selection mechanism is designed to capture the asymmetric inter-agent dependencies overlooked by homogeneous approaches.
- We design a structured curriculum learning strategy that systematically mitigates the risks of sparse rewards and catastrophic forgetting. By combining stage-proportional experience replay with a hierarchical policy transfer mechanism, our framework enables stable and efficient learning across tasks of increasing complexity.

The remainder of this paper is organized as follows. Section 2 presents the problem statement, including the system model and formulation. In Section 3, the proposed AC-MASAC framework is detailed, covering both the heterogeneous actor-critic architecture and the structured curriculum learning strategy. In Section 4, the effectiveness of the proposed method is validated through comprehensive simulation experiments and ablation studies. Finally, Section 5 provides the conclusion.

2 PROBLEM STATEMENT

This paper addresses the problem of cooperative path planning for a heterogeneous multi-UAV swarm operating in a two-dimensional environment with dynamic obstacles. This problem is formally modeled as a Partially Observable Markov Decision Process (POMDP) [10], which provides a rigorous framework for sequential decision-making under uncertainty. The swarm is composed of a designated Leader agent and multiple Follower agents. The primary objective for the Leader is to navigate from a starting position to a target location while avoiding collisions. The primary objective for the Followers is to maintain a specified formation relative to the Leader's trajectory.

To formulate this sequential decision-making problem for a learning-based solution, we define the core components of the agent-environment interface. Each agent i perceives a local observation z_i based on its role. The Leader's observation is a vector $z_i^{(L)} = [x_i, y_i, v_i, \psi_i, x_G, y_G, O_{\text{flag}}]$, which includes its position (x_i, y_i), speed v_i, heading angle ψ_i, the target's coordinates (x_G, y_G), and a binary obstacle detection flag. In contrast, a Follower's observation is $z_i^{(F)} = [x_i, y_i, v_i, \psi_i, x_L, y_L, v_L]$, which contains its own state alongside the Leader's position (x_L, y_L) and speed v_L. The action space for each agent is continuous, where an action $a = [a_c, \omega_c]$ consists of a linear acceleration command a_c and an angular velocity command ω_c, each bounded by the UAV's specific kinematic limits. The performance of any policy aimed at solving this problem

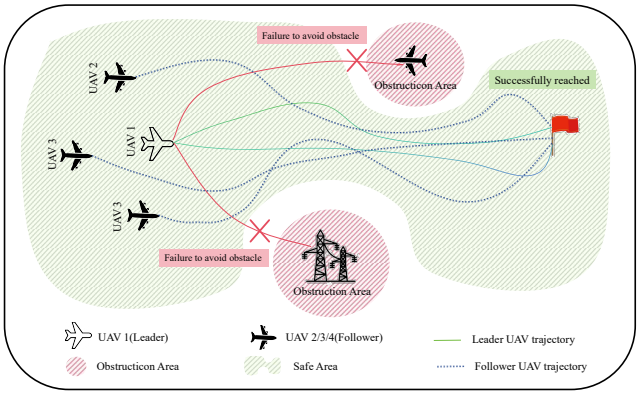


Figure 1: Conceptual diagram of the multi-UAV cooperative path planning task. The swarm, composed of a Leader and multiple Followers, must navigate towards a target while avoiding obstacles and maintaining formation. Each UAV makes decentralized decisions based on its role-specific local observations.

is evaluated using three key metrics. The Success Rate (SR) is defined as the proportion of episodes in which the Leader successfully reaches the target without collision. The Formation Keeping Rate (FKR) measures coordination quality, calculated as the proportion of time steps during which Followers successfully maintain their specified formation constraints. Finally, the Success-weighted Mission Time (SMT), defined as the average episode length multiplied by the Success Rate, is used to evaluate the overall efficiency of the generated solution.

3 AC-MASAC APPROACH

3.1 POMDP Formulation

The cooperative path planning problem under partial observability is modeled as a decentralized Partially Observable Markov Decision Process (POMDP) [15, 23], an extension of the traditional Markov Decision Process [2, 3]. The POMDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, O, \gamma \rangle$. Here, $\mathcal{S}$ represents the set of true environmental states, which are not directly observable by the agents. $\mathcal{A}$ is the joint action set composed of individual agent actions, which are known to the agents. $\mathcal{P}$ is the state transition function, governed by the environment dynamics and generally unknown to the agents. $\mathcal{R}$ is the reward function. $\mathcal{Z}$ is the set of joint observations, from which each agent draws its local view. O is the observation function that maps states to observations, which is also unknown. Finally, γ is the discount factor.

State Space ($\mathcal{S}$): Following the Centralized Training with Decentralized Execution (CTDE) paradigm, the global state $s_t \in \mathcal{S}$ is defined as the concatenation of all agents' local observations: $s_t = \{z_t^{(L)}, z_t^{(F_1)}, \ldots, z_t^{(F_N)}\}$. This global state serves as the input for the centralized critic to estimate the joint value function based on the swarm's collective perception.

Observation Space ($\mathcal{Z}$): The observation for the i-th agent, $z_i \in \mathcal{Z}_i$, consists of its own flight status and role-specific information. For the Leader UAV, the observation vector is $z_i^{(L)} = [x_i, y_i, v_i, \psi_i, x_G, y_G, O_{\text{flag}}]$, where O_{flag} is a binary indicator set to 1 if the nearest obstacle is within 40m, and 0 otherwise. For a Follower UAV, its observation vector is $z_i^{(F)} = [x_i, y_i, v_i, \psi_i, x_L, y_L, v_L]$, containing information about the Leader. The observation function O is subject to simulated sensor noise, modeled as an additive Gaussian process $\mathcal{N}(0, \sigma^2_{\text{noise}})$, where $\sigma_{\text{noise}} = 0.1$.

Action Space ($\mathcal{A}$): The action for the i-th agent, $a_i \in \mathcal{A}_i$, is a continuous two-dimensional vector $a_i = [u_i, \omega_i]^T$, representing linear acceleration and angular velocity. These actions are constrained by heterogeneous kinematics: the Leader UAV has a velocity range of [10, 20] m/s and acceleration $|u_i| \leq 3\,\text{m/s}^2$, while Followers have a velocity range of [10, 30] m/s and acceleration $|u_i| \leq 6\,\text{m/s}^2$.The angular velocity for all agents is constrained to $\omega_i \in [-\pi/3, \pi/3]$ rad/s.

Transition Function ($\mathcal{P}$): The state transition function $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ is deterministic and governed by the discretized UAV dynamics model[4], with a simulation time step of $\Delta t = 0.1s$:

$$\begin{cases} x_i(t+1) = x_i(t) + v_i(t)\Delta t \cos\psi_i(t) \\ y_i(t+1) = y_i(t) + v_i(t)\Delta t \sin\psi_i(t) \\ v_i(t+1) = v_i(t) + u_i(t)\Delta t \\ \psi_i(t+1) = \psi_i(t) + \omega_i(t)\Delta t \end{cases} \tag{1}$$

Reward Function ($\mathcal{R}$): Role-specific reward functions are designed to guide the agents towards their respective objectives. The total reward for each agent is the sum of several components, and the discount factor is denoted by γ.

Leader's Reward (R_t^L):

Goal Achievement Reward (r_g): A significant sparse reward is provided upon reaching the target. To mitigate the challenge of this sparse signal, a dense reward shaping component based on the negative Euclidean distance to the goal, denoted as d_g, is also included.

$$r_g = \begin{cases} 1000.0, & \text{if } d_g < 40 \\ -0.001 \cdot d_g, & \text{otherwise} \end{cases} \tag{2}$$

Obstacle Penalty (r_o): To ensure a safe distance from obstacles, a staged penalty is imposed when the Euclidean distance d_o to the nearest obstacle falls below a certain threshold.

$$r_o = \begin{cases} -500.0, & \text{if } d_o < 20 \\ -2.0, & \text{if } 20 \leq d_o \leq 40 \\ 0.0, & \text{otherwise} \end{cases} \tag{3}$$

Boundary Penalty (r_e): A penalty is applied when the Manhattan distance d_b to the nearest boundary is less than 50.

$$r_e = \begin{cases} -1.0, & \text{if } d_b < 50 \\ 0.0, & \text{otherwise} \end{cases} \tag{4}$$

Formation Distance Penalty (r_{f1}): A linear penalty is imposed when the maximum distance to any follower, $\max_j(d_{L,j})$, exceeds the ideal formation distance of 50.

$$r_{f1} = \begin{cases} 0.0, & \text{if } \max_j(d_{L,j}) \leq 50 \\ -0.001 \cdot \max_j(d_{L,j}), & \text{otherwise} \end{cases} \tag{5}$$

Velocity Matching Reward (r_{s1}): A reward is given for each follower where the speed difference with the leader is less than 1. The total reward is the sum of these individual rewards over all N followers.

$$r_{s1} = \sum_{i=1}^{N} \begin{cases} 1.0, & \text{if } |v_L - v_{F,i}| < 1 \\ 0.0, & \text{otherwise} \end{cases} \tag{6}$$

The total reward for the Leader is the sum of these components: $R_t^L = r_g + r_o + r_e + r_{f1} + r_{s1}$.

Follower's Reward (R_t^F):

Formation Distance Reward (r_{f2}): A staged reward is given when the distance d_L to the leader is within the ideal range; otherwise, a linear penalty is imposed.

$$r_{f2} = \begin{cases} 100.0, & \text{if } 0 < d_L < 40 \\ 10.0, & \text{if } 40 \leq d_L < 50 \\ -0.01 \cdot d_L, & \text{otherwise} \end{cases} \tag{7}$$

Velocity Matching Reward (r_{s2}): When within the ideal formation distance and the speed difference is less than 1, a velocity matching reward is given.

$$r_{s2} = \begin{cases} 100.0, & \text{if } d_L < 50 \text{ and } |v_L - v_i| < 1 \\ 0.0, & \text{otherwise} \end{cases} \tag{8}$$

The total reward for the Follower is the sum of its components: $R_t^F = r_{f2} + r_{s2}$.

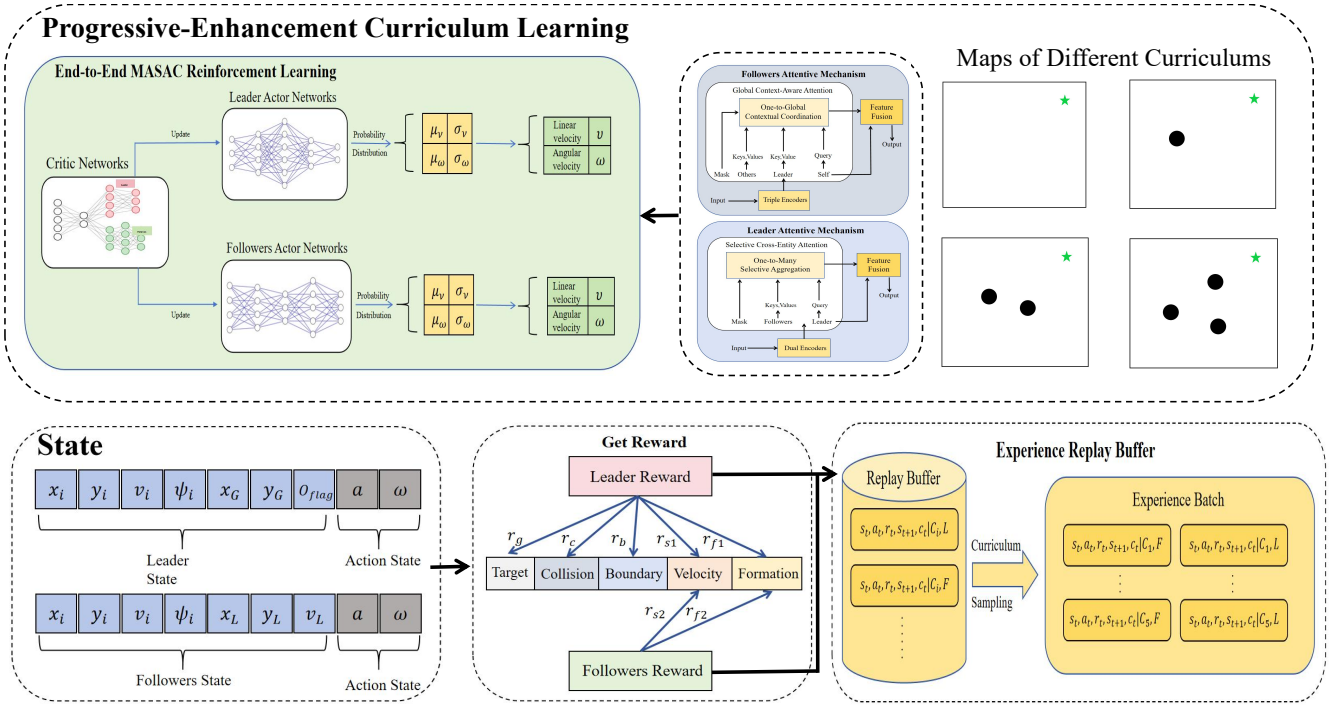


Figure 2: AC-MASAC framework overview. *Top*: Curriculum learning module controlling task difficulty. *Bottom*: Attention-enhanced actor-critic networks processing states for action generation and training via a curriculum-managed replay buffer.

An episode terminates when the Leader reaches the target, or when any agent collides with an obstacle.

3.2 Heterogeneous Actor-Critic Architecture

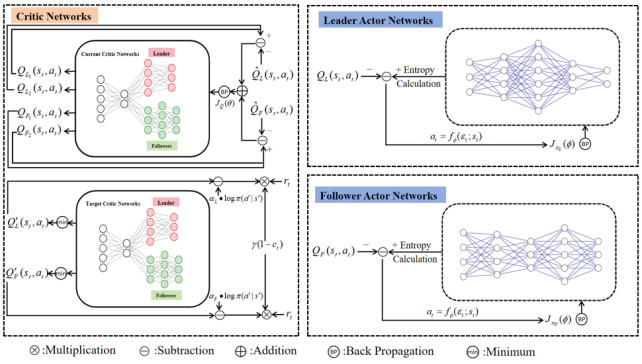


Figure 3: AC-MASAC architecture diagram.

The AC-MASAC algorithm is built upon the Soft Actor-Critic (SAC) framework [6], which optimizes a policy π to maximize an entropy-regularized objective function. The objective is to find a policy that balances the maximization of cumulative reward with the maximization of its own entropy, which encourages exploration and enhances robustness:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right] \tag{9}$$

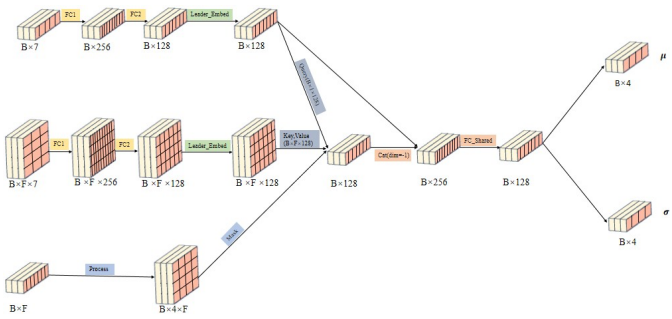
The temperature parameter, α, serves as a trade-off coefficient. Instead of being a fixed hyperparameter, α is learned automatically by minimizing the following loss function, which aims to maintain a target entropy level $\bar{\mathcal{H}}$:

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi_t} \left[-\alpha \log \pi_t(a_t|s_t) - \alpha\bar{\mathcal{H}} \right] \tag{10}$$

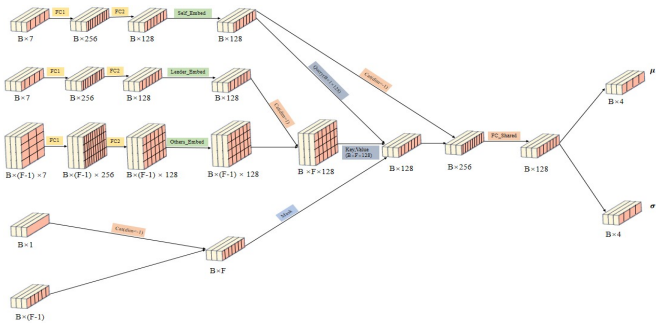
The core of our contribution lies in the design of a heterogeneous actor-critic architecture, termed AC-MASAC. Its overall framework is depicted in Figure 2, while the detailed network architecture is shown in Figure 3. This approach explicitly models the Leader-Follower roles, directly addressing the limitations of the homogeneity assumption identified in the introduction. Instead of using a single, monolithic policy structure, we introduce two distinct, role-specific actor networks and a critic whose structure mirrors this heterogeneity. This design provides an inductive bias that facilitates the learning of specialized, coordinated behaviors, overcoming the challenge of unstructured state representations in prior multi-agent actor-critic methods.

Heterogeneous Actor Networks as illustrated in Figure 4, the architecture employs two distinct actor networks, whose designs are tailored to the specific informational needs of each role.

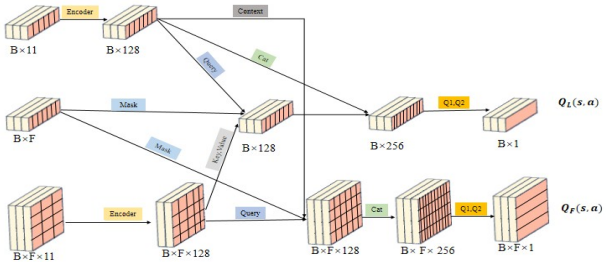
The Leader Actor, whose architecture is detailed in Figure 4(a), utilizes a selective cross-entity attention mechanism. The input is the Leader's local observation $z_i^{(L)}$. It is first passed through an



(a) Leader Actor network architecture



(b) Follower Actor network architecture



(c) Structured Attention Critic network architecture

Figure 4: Attention-based network architectures. (a) Leader Actor, (b) Follower Actor, and (c) Structured Attention Critic. Inputs are role-specific states; outputs are Gaussian policy parameters (μ, σ).

embedding layer ϕ_e^L to produce an embedding e^L, which serves as the query (Q). The observations of the followers $\{z_j^{(F)}\}$ are similarly embedded via a shared embedding layer ϕ_e^F to produce keys (K) and values (V). To handle a variable number of followers and partial observability, a head-wise attention mask is applied, which allows the network to ignore non-existent or out-of-range agents during the attention score calculation. The attention mechanism then computes a context vector representing the aggregated follower states.

The Follower Actor(Figure 4(b)) uses a global context-aware attention mechanism. Its local observation $z_i^{(F)}$ is embedded via ϕ_e^F to form the query (Q). The context is formed by the Leader's observation $z_i^{(L)}$ and all other followers' observations $\{z_j^{(F)}\}_{j\neq i}$, which are embedded to provide the keys (K) and values (V). To improve sample efficiency, the Follower Actor networks share parameters across all follower agents.

For both actor networks, the final processed features are projected to produce the parameters of a Gaussian policy, yielding a mean μ and a log standard deviation $\log\sigma$ that define the distribution from which continuous actions are sampled.

Structured Attention Critic to maintain policy-value consistency within the CTDE framework, the critic's architecture, shown in Figure 4(c), mirrors the actor's heterogeneous attention structure. It employs a twin Q-network design to mitigate overestimation bias. Each Q-network contains two parallel branches: a *leader branch* that computes the Q-value for the Leader by applying selective attention over follower state-action pairs, and a *follower branch* that computes Q-values for followers using global context attention. During training, the critic receives the global state s_t (approximated by the set of all observations $\{z_t^i\}$) and the joint action a_t to learn a centralized value function. The target Q-value, used for the Bellman update, is computed as the minimum of the two target Q-network outputs. The target networks are updated via a soft update from their main network counterparts: $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$ with a small update rate τ.

For the implementation, learning rates were set to 1×10^{-4} for the actor networks and 3×10^{-4} for the critic networks. The discount factor γ was 0.99, the experience buffer size was 5×10^4, and the batch size was 256. The soft update rate was $\tau = 0.01$. The Adam optimizer was used for all networks. Within each Multi-Head Attention block, the embedding dimension was $E = 128$, the number of heads was $H = 4$, and a dropout probability of 0.1 was applied to prevent overfitting.

3.3 Structured Curriculum Learning Framework

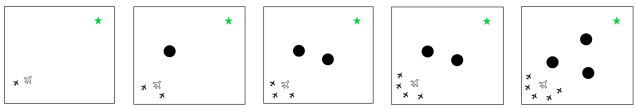


Figure 5: Environment for different levels of training

The AC-MASAC algorithm incorporates a curriculum learning framework to structure the training process. The agent is trained on a predefined sequence of tasks, $C = (T_1, T_2, \ldots, T_K)$, ordered by progressively increasing complexity. Each task T_k is defined by a set of environmental parameters $\xi_k = (N_{L,k}, N_{F,k}, N_{O,k})$, representing the number of leaders, followers, and obstacles, respectively, where complexity increases such that $N_{F,k+1} \geq N_{F,k}$ and $N_{O,k+1} \geq N_{O,k}$. To foster policy generalization and prevent overfitting to specific initial configurations, the starting conditions for each episode are randomized. This includes the initial states of the leader and all follower agents, as well as the locations of the goal and all dynamic obstacles, which are sampled from predefined distributions within

the operational area. This systematic escalation in environmental difficulty ensures the agent builds a foundation of skills before confronting the full complexity of the final task, as illustrated in Figure 5.

Algorithm 1 Progressive Curriculum Learning for AC-MASAC

Initialize MASAC controller M, curriculum manager CM, stage $s \leftarrow 1$, knowledge transfer $KT \leftarrow \varnothing$
for $s = 1$ to max_stages **do**
  Task$_s \leftarrow$ generate_fixed_task(difficulty$_s$)
  if $s > 1$ **then**
    $M \leftarrow KT$.transfer(M, agent_count$_s$)
  end if
  success_rate $\leftarrow 0$, episode $\leftarrow 0$
  while success_rate $< \theta$ and episode $<$ max_episodes **do**
    $E \leftarrow$ Task$_s$.create_environment()
    ratio $\leftarrow$ adaptive_sampling_ratio(s, episode)
    M.set_replay_ratio(ratio)
    M.train(current_stage $= s$, sampling_ratio $=$ ratio)
    success_rate $\leftarrow$ evaluate_performance(M, E)
    episode $\leftarrow$ episode + 1
  end while
  $KT \leftarrow M$.export_parameters()
end for

The sequencing of tasks and the criteria for switching between them follow a deterministic linear progression through the task sequence C. The transition from task T_k to T_{k+1} is triggered only when the agent's performance, evaluated over a sliding window of W episodes, meets a set of predefined thresholds. These criteria are based on the metrics defined in Section 2: a success rate $SR \geq \theta_{SR}$, a reward coefficient of variation $CV_R \leq \theta_{CV}$, and a formation keeping rate $FKR \geq \theta_{FKR}$.

A hierarchical knowledge transfer mechanism is applied to the network parameters to leverage previously learned policies upon switching to a new task T_{k+1}. Let θ_A and ϕ_C denote the parameters of the actor and critic networks, respectively. Specifically, the Leader's actor policy is fully transferred ($\theta_{A,L}^{(k+1)} \leftarrow \theta_{A,L}^{(k)}$), as its core objective remains consistent across tasks. For the Follower actors, parameters for the $N_{F,k}$ existing agents are preserved ($\theta_{A,F_i}^{(k+1)} \leftarrow \theta_{A,F_i}^{(k)}$ for $i = 1, \ldots, N_{F,k}$), while any newly introduced followers ($j > N_{F,k}$) have their parameters replicated from the most experienced existing follower ($\theta_{A,F_j}^{(k+1)} \leftarrow \theta_{A,F_{N_{F,k}}}^{(k)}$) to bootstrap coordination behavior. In contrast, the critic networks are re-initialized ($\phi_C^{(k+1)} \sim \text{Init}()$), allowing the value function to adapt to the new task's specific dynamics and reward landscape without being biased by outdated value estimations.

To mitigate catastrophic forgetting, a stage-proportional experience sampling mechanism manages the experience replay buffer $\mathcal{D}$ as a collection of stage-specific sub-buffers, $\mathcal{D} = \bigcup_{j=1}^{k} \mathcal{D}_j$. During training at stage k, the proportion of experiences sampled from historical stages ($j < k$), denoted $\rho_{\text{old}}(k)$, is determined by a stage-dependent decay function:

$$\rho_{\text{old}}(k) = \max(\rho_{\text{base}} - \eta \cdot (k-1), \rho_{\text{min}}) \tag{11}$$

where ρ_{base} is the initial proportion of historical experiences, η is the per-stage decay rate, and ρ_{min} is the minimum proportion. A training mini-batch is thus composed of samples drawn from the current stage buffer $\mathcal{D}_k$ and the historical buffer $\mathcal{D}_{\text{hist}} = \bigcup_{j=1}^{k-1} \mathcal{D}_j$ according to this ratio. This mechanism maintains a balance between knowledge retention and adaptation to the current task. In our implementation, the hyperparameters are set as follows: $\theta_{SR} = 0.9$, $\theta_{CV} = 0.3$, $\theta_{FKR} = 0.8$, $W = 15$, $\rho_{\text{base}} = 0.8$, $\eta = 0.1$, and $\rho_{\text{min}} = 0.2$.

4 EXPERIMENTS AND DISCUSSION

To comprehensively evaluate the performance of the proposed AC-MASAC algorithm, we compared it with two mainstream baseline methods. For the non-learning-based approach, we utilized RRT* [11] combined with a Pure Pursuit Controller, which uses complete global information to generate a centralized reference path, serving as a static planning benchmark. For learning-based methods, we chose MASAC [4] and MADDPG [14] as comparative baselines. All hyperparameters for the baseline algorithms were set to the default recommended values from their official open-source implementations.

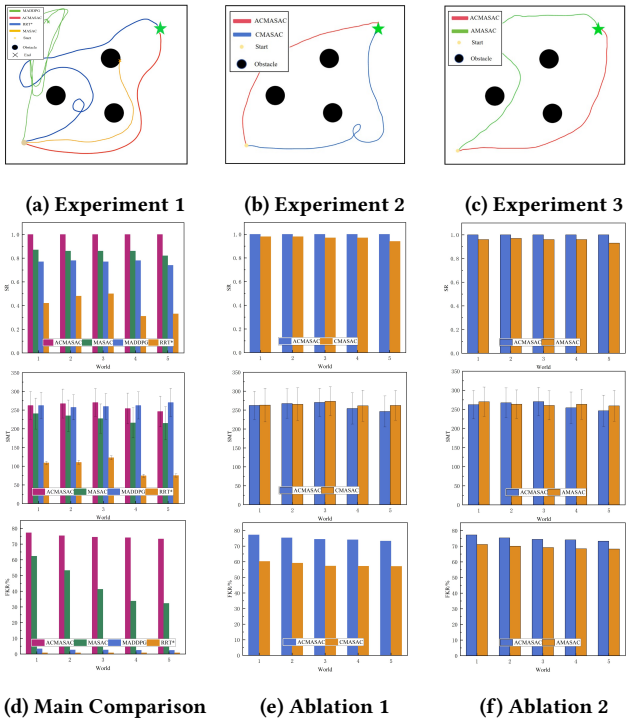


Figure 6: Experimental trajectory diagram, SR, SMT and FKR, (a)-(c) show the leader trajectory plots from three different experiments in simulation environment world5. (d)-(f) show Performance Results plots in various test worlds, including SR, SMT and FKR. Histogram means the average value for different metrics and the error bar is the range under a 95% confidence interval.

4.1 Simulation Experiment

The simulation environment, custom-developed based on the OpenAI Gym framework with Pygame for visualization, simulates a 700×600-meter airspace and is designed to model heterogeneous agent dynamics and dynamic obstacle configurations.

To evaluate generalization, a suite of test environments with varying difficulty levels was used, as depicted in Figure 5. Each algorithm was tested for 100 episodes per environment, with each episode lasting a maximum of 1000 iterations and initiated with randomized agent positions, target locations, and obstacle placements. The performance of four algorithms AC-MASAC, MASAC, MADDPG, and RRT*—were recorded based on three key metrics: Success Rate (SR), Success-weighted Mission Time (SMT), and Formation Keeping Rate (FKR).

As shown in the representative trajectories in Figure 6(a), AC-MASAC consistently generated a smooth and direct path to the target, benefiting from its attention-based coordination and curriculum-honed policy. The MADDPG algorithm often produced hesitant or circuitous routes, indicating struggles with multi-agent credit assignment. The MASAC algorithm, while better than MADDPG, still showed inefficiencies in obstacle-dense areas. The RRT* algorithm, being a non-learning method, found feasible paths but they were frequently suboptimal and lacked the fluid dynamics of the learned policies.

The quantitative results in Figure 6(d) further support these observations. Across all test worlds, AC-MASAC achieved the highest SR and, most critically, a vastly superior FKR, demonstrating its robust coordination capabilities. While MASAC and MADDPG showed reasonable performance in simpler worlds, their success rates and formation stability degraded significantly as complexity increased, indicating poorer generalization. The RRT* method, while consistent, scored lowest on efficiency (SMT) and had a near-zero FKR as it lacks an inherent coordination mechanism. Overall, while all learning-based methods demonstrated flexibility, AC-MASAC exhibited the best all-around performance. The significant lead in FKR explicitly validates the effectiveness of our heterogeneous attention mechanism in modeling Leader-Follower dynamics, while the superior SR and SMT in complex environments underscore the contribution of the structured curriculum framework in achieving robust policy convergence where other methods falter.

4.2 Ablation Experiment

1) Attention Mechanism. To validate the efficacy of the proposed attention mechanism, we conducted an ablation study comparing our final AC-MASAC algorithm against a variant without the attention mechanism, hereafter referred to as C-MASAC (Curriculum-only MASAC). Both models were trained and tested under identical settings.

As illustrated in the representative path comparison in Figure 6(b), AC-MASAC navigates around obstacles while maintaining a smooth and goal-oriented trajectory. In contrast, C-MASAC, lacking dynamic awareness of teammate states, follows a path that appears more tortuous and hesitant, particularly at critical junctures requiring collective decision-making. AC-MASAC effectively leverages the attention mechanism to dynamically weigh the importance of information from different teammates, leading to more cooperative decisions that directly manifest as tighter formations and more optimal path choices.

Furthermore, the quantitative metrics in Figure 6(e) show that AC-MASAC's Formation Keeping Rate (FKR) consistently surpassed that of C-MASAC by an average of over 15% across all test environments. This provides strong evidence for the central role of the attention mechanism in enhancing multi-agent coordination. Concurrently, the higher Success Rate (SR) indicates that improved coordination translates directly into a greater probability of mission success.

2) Curriculum Learning. To evaluate the role of curriculum learning in enhancing algorithm generalization and final performance convergence, we conducted a comparative experiment between our full AC-MASAC algorithm and A-MASAC (Attention-only MASAC), an attention-based variant trained directly in the most complex environment without the structured curriculum. As shown in the reward curve in Figure 7, A-MASAC experienced significant reward fluctuations during the initial training phase due to the immediate exposure to a complex task, and its final converged reward level was notably lower than that of AC-MASAC. This suggests that learning a complex multi-agent coordination policy from scratch is exceedingly difficult.

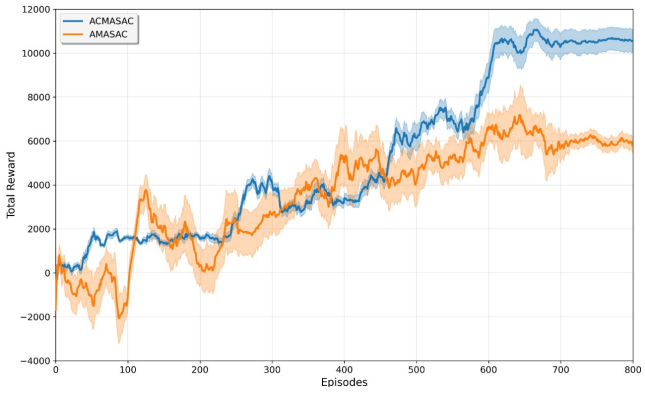


Figure 7: Comparison of training reward curves for AC-MASAC and A-MASAC.

Conversely, AC-MASAC, through its simple-to-complex curriculum learning paradigm, exhibited a more stable learning curve and converged to a significantly higher performance level, surpassing A-MASAC's final average reward by approximately 82%. This highlights the importance of curriculum learning in establishing foundational policies and avoiding local optima. The path comparisons and quantitative metrics in Figures 6(c) and 6(f) further corroborate these findings: AC-MASAC consistently outperformed A-MASAC in both Success Rate (SR) and Formation Keeping Rate (FKR), indicating that the knowledge acquired through curriculum learning generalizes more effectively, enabling superior performance in a variety of complex environments.

3) Robustness to Communication Imperfections. While the primary experiments presented above assume ideal communication channels, real-world deployment of UAV swarms inevitably involves non-ideal conditions such as packet loss and latency. To evaluate the robustness of the proposed AC-MASAC against such practical challenges, we conducted additional tests introducing stochastic packet dropout during the decentralized execution phase. In these scenarios, each agent fails to receive state updates from a specific teammate with a predefined probability p_{drop} at each time step. Qualitative results derived from complex test environments indicate that AC-MASAC maintains resilient coordination performance under moderate packet loss rates (e.g., $p_{\text{drop}} \approx 10 - 20\%$). Although formation precision naturally exhibits graceful degradation as p_{drop} increases beyond this range, the swarm successfully avoids catastrophic failures such as inter-agent collisions or complete formation breakup. This observed robustness can be largely attributed to the inherent properties of the proposed heterogeneous attention mechanism, which dynamically learns to down-weight unreliable or missing signals and refocus on available critical information, thereby sustaining effective cooperative decision-making under uncertainty.

5 CONCLUSION

This paper addresses the cooperative path planning control of a heterogeneous UAV swarm using a multi-agent reinforcement learning method. By defining a Leader-Follower role structure, we established a decentralized Partially Observable Markov Decision Process (POMDP) that takes into account both individual agent dynamics and multi-agent coordination objectives. A heterogeneous actor-critic architecture was developed to address the role-specific decision-making requirements. Within this architecture, a role-aware attention mechanism was designed to process the unstructured state information inherent in multi-agent systems. The constructed system learns distinct policies for Leader and Follower agents based on their differentiated observation and action spaces.

To address the challenges of training instability and slow convergence in complex, sequential tasks, a structured curriculum learning framework was integrated. Based on a sequence of tasks with progressively increasing complexity, the framework utilizes a hierarchical knowledge transfer mechanism for policy initialization and a stage-proportional experience replay strategy to mitigate catastrophic forgetting. Based on this framework, a robust, curriculum-driven policy was synthesized, which adapts to environments of varying difficulty.

The proposed AC-MASAC method is applicable to cases with heterogeneous agent roles and complex, dynamic environments, and it achieves superior performance in success rate and formation keeping compared with conventional MARL baselines. In the future, the AC-MASAC framework will be extended to address the challenges of sim-to-real transfer for deployment on physical hardware platforms.

REFERENCES

[1] Johannes Ackermann, Volker Gabler, Takayuki Osa, and Masashi Sugiyama. 2019. Reducing overestimation bias in multi-agent domains using double centralized critics. *arXiv preprint arXiv:1910.01465* (2019).

[2] Richard Bellman. 1957. A Markovian decision process. *Journal of mathematics and mechanics* (1957), 679–684.

[3] Dimitri Bertsekas. 2019. *Reinforcement learning and optimal control*. Vol. 1. Athena Scientific.

[4] CL Fang, FS Yang, and Q Pan. 2024. Multi-UAV collaborative path planning based on multi-agent soft actor critic. *Sci. Sin. Inf* 54 (2024), 1871–1883.

[5] J Foerster. 2018. *Deep multi-agent reinforcement learning*. Ph.D. Dissertation. University of Oxford.

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. Pmlr, 1861–1870.

[7] Jiale Han, Yi Zhu, and Jian Yang. 2025. A Deep Reinforcement Learning Method for Collision Avoidance with Dense Speed-Constrained Multi-UAV. *IEEE Robotics and Automation Letters* (2025).

[8] Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, Angel Romero, Robert Penicka, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza. 2024. Autonomous drone racing: A survey. *IEEE Transactions on Robotics* 40 (2024), 3044–3067.

[9] Shariq Iqbal and Fei Sha. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 2961–2970.

[10] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence* 101, 1-2 (1998), 99–134.

[11] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. 2011. Anytime motion planning using the RRT. In *2011 IEEE international conference on robotics and automation*. ieee, 1478–1483.

[12] Oussama Khatib. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research* 5, 1 (1986), 90–98.

[13] Chengju Liu, Junqiang Han, and Kang An. 2017. Dynamic path planning based on an improved RRT algorithm for RoboCup robot. *Robot* 39, 1 (2017), 8–15.

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

[15] Shankarachary Ragi and Edwin KP Chong. 2013. UAV path planning in a dynamic environment via partially observable Markov decision process. *IEEE Trans. Aerospace Electron. Systems* 49, 4 (2013), 2397–2412.

[16] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research* 21, 178 (2020), 1–51.

[17] Zhuang Shao, Fei Yan, Zhou Zhou, and Xiaoping Zhu. 2019. Path planning for multi-UAV formation rendezvous based on distributed cooperative particle swarm optimization. *Applied Sciences* 9, 13 (2019), 2621.

[18] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*. PMLR, 5887–5896.

[19] Chuxiong Sun, Zehua Zang, Jiabao Li, Jiangmeng Li, Xiao Xu, Rui Wang, and Changwen Zheng. 2024. T2mac: Targeted and trusted multi-agent communication through selective engagement and evidence-driven integration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15154–15163.

[20] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[21] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.

[22] Hanfu Wang and Weidong Chen. 2022. Multi-robot path planning with due times. *IEEE Robotics and Automation Letters* 7, 2 (2022), 4829–4836.

[23] Liang Wang, Kezhi Wang, Cunhua Pan, Wei Xu, Nauman Aslam, and Lajos Hanzo. 2020. Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking* 7, 1 (2020), 73–84.

[24] Xiaojun Xing, Zhiwei Zhou, Yan Li, Bing Xiao, and Yilin Xun. 2024. Multi-UAV adaptive cooperative formation trajectory planning based on an improved MATD3 algorithm of deep reinforcement learning. *IEEE Transactions on Vehicular Technology* 73, 9 (2024), 12484–12499.