

Article

A Learnable Sparse Attention Graph Architecture for Heterogeneous Multi-UAV Air-to-Ground Mission Planning

Haolun Sun ¹, Xiangke Guo ^{2,*}, Xiangwei Bu ² and Gang Wang ²¹ Graduate School, Air Force Engineering University, Xi'an 710051, China; 18644884707@163.com² Air Defense and Antimissile School, Air Force Engineering University, Xi'an 710051, China; buxiangwei1987@126.com (X.B.); 13206579035@163.com (G.W.)

* Correspondence: guosyanyu@163.com

Highlights

What are the main findings?

- A learnable sparse attention graph architecture (SAGA) is proposed under the MAPPO CTDE framework, integrating a GATv2 multi-head graph encoder, learnable edge predictor and L1 sparsity penalty to jointly address partial observability, heterogeneous role coordination and communication constraints in multi-UAV air-ground missions.
- Comprehensive numerical results demonstrate that SAGA achieves faster reward convergence, higher steady mission reward, more target destruction and lower UAV attrition than MADDPG, MATD3 and QMIX.

What are the implications of the main findings?

- The proposed method provides a generalizable MARL design template for heterogeneous multi-agent adversarial missions, which exhibits cross-scenario transfer robustness.
- The joint optimization of dynamic communication topology and graph attention reasoning breaks the limitation of fully connected and no-communication MARL baselines.

Abstract

In the complex problem of air-to-ground mission planning, multi-UAV systems face significant challenges such as system complexity and heterogeneity, insufficient target observability, and difficulties in collaborating information sharing. To address these issues, this paper proposes a novel learnable sparse attention graph architecture (SAGA). This architecture deeply integrates graph reasoning and policy optimization within the MAPPO framework and includes three innovative mechanisms: (i) a GATv2-based graph neural network encoder that performs multi-round distributed consensus on the communication graph among UAVs via a multi-head attention mechanism, enabling selective aggregation of tactical information; (ii) an edge predictor that learns to prune low-value communication links, generating a sparse and mission-adaptive communication topology; and (iii) an L1 sparsity penalty term that further enhances communication efficiency. In a self-developed simulation environment for heterogeneous multi-UAV mission planning, comprehensive comparative experiments were conducted against the following baseline reinforcement learning algorithms: MADDPG, MATD3, QMIX, MAPPO, TarMAC, DGN, and G2ANet. The experimental results show that SAGA achieves reward values of 390 and 1100 in small-scale and large-scale scenarios, and outperforms the best-

Academic Editor: Diego González-Aguilera

Received: 7 August 2026

Revised: 28 August 2026

Accepted: 8 September 2026

Published: 10 September 2026

Copyright: © 2026 by the author.

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.

performing baseline algorithm by more than 20% across all operational performance metrics. Generalization experiments validate the model's robust transfer capability under unknown defense deployment modes. Ablation experiments further confirmed the individual contributions of the three components. This study provides an innovative and effective method for mission planning of heterogeneous multi-UAV systems in partially observable adversarial environments.

Keywords: multi-UAV; mission planning; air-to-ground mission; SAGA

1. Introduction

With the rapid advancement of unmanned aerial vehicle (UAV) technologies, multi-UAV collaborative operations have emerged as a transformative paradigm in modern military applications. Compared with single-UAV missions, multi-UAV systems offer significant advantages in mission efficiency, survivability, and operational flexibility through collaborated mission planning and information sharing [1,2]. Among various multi-UAV mission types, the air-to-ground mission represents one of the most challenging and significant scenarios, in which a heterogeneous UAV team must collaboratively neutralize surface-to-air missile (SAM) sites to clear safe corridors for subsequent operations [3]. The heterogeneity of the UAV team, which typically consists of reconnaissance, strike, and electronic-warfare platforms with complementary capabilities, further compounds the coordination complexity [4].

The air-to-ground mission poses several fundamental challenges that distinguish it from generic multi-agent collaboration problems. First, each UAV operates under partial observability, its onboard sensors provide only a limited view of the battlefield, and other information must be acquired through inter-UAV communication. Second, the heterogeneous UAV team demands collaboration based on mission roles; however, different UAVs form a tightly coupled, sequential dependency chain, making collaboration strategies for homogeneous formations unsuitable. Third, communication in contested environments is severely constrained, requiring agents to selectively transmit only mission-critical information rather than flooding the network with redundant messages [5]. These challenges call for a decision-making framework that simultaneously addresses partial observability, heterogeneous role collaboration, and communication efficiency in a unified architecture.

Methods widely used to solve UAV mission planning have relied on mathematical optimization [6] and game-theoretic formulations [7]. Optimization-based methods, such as mixed-integer linear programming (MILP) and heuristic search algorithms, can produce high-quality plans for small-scale, static scenarios but suffer from the curse of dimensionality as the number of agents and targets grows. Game-theoretic approaches, including differential games and stochastic games, provide principled frameworks for adversarial reasoning but typically require accurate opponent models and simplified state representations that are difficult to obtain in realistic battlefield environments. More fundamentally, both paradigms assume centralized computation with full state knowledge, making them ill-suited for the decentralized, partially observable nature of real operations [8].

Recent advances in deep reinforcement learning (DRL) have opened new avenues for autonomous decision-making in complex sequential tasks [9]. The extension of DRL to multi-agent settings—multi-agent reinforcement learning (MARL)—has demonstrated remarkable success in cooperative and competitive scenarios, from StarCraft II micromanagement to autonomous driving [10,11]. The centralized training with decentralized execution (CTDE) paradigm enables agents to access global information during training

while relying solely on local observations during execution [12]. Representative CTDE algorithms include MADDPG [13], which extends the actor–critic framework to multi-agent domains; QMIX [14], which factorizes the joint action-value function under a monotonicity constraint; and MAPPO [15], which adapts the highly successful proximal policy optimization (PPO) algorithm to multi-agent settings with demonstrated state-of-the-art performance across a range of cooperative benchmarks.

Despite these advances, directly applying existing CTDE algorithms to the heterogeneous multi-UAV air-to-ground mission planning problem reveals several critical limitations:

- (1) Most MARL algorithms assume either fully independent agents with no explicit communication, or fully connected communication graphs where every agent broadcasts to all others [16]. The former fails to exploit the information-sharing potential of networked UAVs, while the latter generates a large number of irrelevant messages.
- (2) Standard MARL policy networks, typically implemented as shared multi-layer perceptron (MLP), lack the relational inductive bias needed to model the rich interaction patterns among heterogeneous agents [17]. Graph neural networks (GNNs), by representing agents as nodes and communication links as edges in a dynamic graph, offer a natural framework for learning relational representations in multi-agent systems [18]. However, the integration of GNNs with learnable, sparse communication topologies in the context of heterogeneous multi-UAV air-to-ground mission planning remains largely unexplored.

To address these gaps, we propose a sparse attention graph architecture (SAGA) for heterogeneous multi-UAV air-to-ground mission planning. SAGA integrates three components within MAPPO: (1) a GATv2-based graph neural network encoder that performs multi-round distributed consensus via multi-head attention over the inter-UAV communication graph, enabling each agent to selectively aggregate tactical information from its communication neighbors; (2) an edge predictor, jointly trained with the policy, that learns to prune low-value communication links and generate a sparse, task-adaptive communication topology; and (3) an L1 sparsity penalty on the predicted edge probabilities that explicitly incentivizes communication efficiency. We evaluate SAGA on a self-built simulation environment and benchmark it against three established MARL algorithms: MADDPG, MATD3, and QMIX. Comparisons were conducted across various scenarios to evaluate the algorithms' overall performance and generalizability. The results of the comparative experiments show that, in terms of both algorithmic performance and operational metrics, the SAGA algorithm demonstrates stable performance and significantly outperforms all comparison benchmarks. Generalization experiments confirm that policies trained in standard scenarios transfer robustly to unseen configurations. Ablation studies further validate the individual contributions of the three components.

The main contributions of this paper are summarized as follows:

- (1) We formalize heterogeneous multi-UAV air-to-ground mission planning as a Dec-POMDP with hybrid continuous–discrete action spaces, modeling the partial observability, role heterogeneity, and complex action constraints inherent to realistic operations.
- (2) We propose SAGA, a sparse-attention graph architecture that integrates GATv2-based graph message passing, an MLP edge predictor for learned sparse topology, and L1 sparsity regularization into a unified MAPPO-based CTDE framework. Additionally, a hybrid action space is designed, processed through parallel policy heads, enabling continuous maneuver control alongside discrete tactical.
- (3) Through extensive experiments, generalization experiments, and comprehensive ablation experiments under simulation scenarios of varying scales and defense modes,

we demonstrate SAGA's superior performance and generalization capability, and the individual contribution of each architectural component, providing actionable insights for designing MARL system architectures for multi-UAV air-to-ground mission planning.

The remainder of this paper is organized as follows. Section 2 reviews related work on mission planning and multi-agent reinforcement learning methods. Section 3 formulates the heterogeneous multi-UAV air-to-ground mission planning problem as a Dec-POMDP. Section 4 describes the system model. Section 5 presents the proposed SAGA algorithm in detail. Section 6 reports experimental results. Section 7 concludes the paper and discusses future research directions.

2. Related Work

Since the widespread adoption of UAVs in the military, the problem of mission planning has remained a central research topic. Research on UAV mission planning typically focused on deterministic optimization formulations. Zhang et al. [19] used the PSO-BAS to plan the spatial 3D route, and it effectively reduced the generated path redundancy. While these optimization-based approaches can produce globally optimal solutions for small instances, they rely on complete and accurate models of the operational environment and scale poorly with the number of agents and mission phases, limiting their applicability to the dynamic, partially observable conditions characteristic of real operations. Game-theoretic methods have been explored as an alternative paradigm for UAV mission planning. Based on the theory of non-cooperative games, Jiao et al. [20] proposed a novel control strategy that enables UAVs to autonomously plan optimal flight trajectories while avoiding conflicts with other UAVs. However, game-theoretic approaches generally require tractable state and action representations, which necessitates significant abstraction of the battlespace. The assumption of rational, optimal opponent behavior may also not hold against adaptive or unpredictable air defense operators. Furthermore, these methods typically assume centralized computation, making them incompatible with the decentralized execution requirements of multi-UAV systems operating in communication-degraded environments.

The application of reinforcement learning to multi-UAV scenarios has gained substantial momentum in recent years. Yang et al. [21] proposed a task-scheduling algorithm based on reinforcement learning, which enables the UAV to adjust its task strategy automatically and dynamically using its calculation of task performance efficiency. Yang et al. [22] proposed NGASAC, a multi-UAV cooperative search algorithm combining normalizing flows with multi-head graph attention, for partially observable low-altitude environments, demonstrating improved search efficiency over non-graph-based baselines. Xu et al. [23] introduced CaST-MASAC, which integrates causal inference and spatio-temporal attention mechanisms for multi-UAV cooperative task planning, showing that causal experience replay can accelerate policy convergence. Qie et al. [24] developed a multi-UAV cooperative encirclement method using SA-DSM-MADDPG with dual stochastic mechanisms for enhanced exploration. Zhang et al. [25] employed MADDPG for multi-UAV cooperative air combat maneuver decisions, while Guo et al. [26] proposed a Hierarchical Hybrid Action Proximal Policy Optimization (HHAPPO) algorithm that addresses the complex optimization challenges of the deep coupling between tactical maneuvering and attack decision. A common limitation across these works is the reliance on either fully connected communication topologies or the absence of explicit inter-agent communication modeling, both of which are suboptimal for environments. Additionally, most existing approaches do not address the role heterogeneity, where different UAV types must perform qualitatively different subtasks in a coordinated temporal sequence.

Recent work has also explored joint optimization of UAV configuration and mission planning for suppression of enemy air defense (SEAD) scenarios. Li et al. [27] proposed a co-optimization framework that simultaneously determines UAV type allocation and mission routes for heterogeneous SEAD operations, highlighting the coupling between platform heterogeneity and mission effectiveness. Jiang et al. [28] confirmed that the current research field of SEAD mission planning mainly faces challenges in fully simulating the complexity of real-world scenarios and in insufficient autonomous task planning capabilities. Therefore, there is an urgent need for a unified framework that can simultaneously address the following requirements of air-to-ground missions: (i) handling partial observability through structured inter-agent communication, (ii) learning role-aware coordination strategies for heterogeneous UAV teams, and (iii) achieving communication efficiency through learned sparse topology. Our work aims to fill this gap by integrating graph-based message passing with learned communication topology into a coherent MARL framework.

The field of MARL has witnessed rapid progress, driven by breakthroughs in deep learning and the increasing complexity of collaborative decision-making problems. The CTDE paradigm has emerged as a particularly effective framework for cooperative MARL, as it mitigates the non-stationarity problem faced by independent learners while preserving decentralized execution capability. Within this paradigm, several algorithmic families have been established. Algorithms such as MADDPG, VDN [29] and QMIX have achieved efficient decentralized execution while maintaining centralized value learning. More recently, MAPPO has demonstrated that the on-policy PPO algorithm, when properly tuned with centralized value functions and parameter sharing, achieves competitive or superior performance to off-policy methods on the StarCraft II multi-agent challenge (SMAC) benchmark. MAPPO's stability, sample efficiency, and compatibility with hybrid action spaces make it a natural choice for complex, long-horizon cooperative tasks such as mission planning.

Graph neural networks have been increasingly adopted in MARL to capture the relational structure inherent to multi-agent systems. The Deep Graph Network (DGN) [30] was among the first to apply graph convolutional networks to multi-agent cooperation, representing agents as graph nodes and using attention-based message passing to aggregate information from neighboring agents. G2ANet [31] introduced a two-stage attention mechanism that first identifies relevant interaction partners before performing message aggregation; MAGIC [32] incorporated a graph attention mechanism guided by the critic's value function to prioritize informative messages; and DICG [33] proposed a deep implicit coordination graph that learns dynamic, task-dependent edge structures. These works have consistently shown that GNN-based architectures outperform MLP-based encoders in cooperative MARL tasks by providing a stronger relational inductive bias. The graph attention network (GAT) [34] computes attention coefficients as a single linear projection of concatenated source–target features followed by LeakyReLU and softmax normalization. Its improved variant GATv2 [35] introduces a more expressive joint nonlinear attention mechanism that conditions on source, target, and edge features, enabling richer inter-agent message weighting. Our SAGA framework adopts GATv2 as the backbone GNN encoder to leverage this enhanced attention expressiveness for heterogeneous UAV collaboration.

Communication learning in MARL addresses the question of what, when, and with whom agents should communicate. Early work in this area includes CommNet [36], which learns continuous communication vectors broadcast by all agents and aggregated via mean pooling, and TarMAC [16], which introduces a targeted, signature-based attention mechanism for selective message reception. IC3Net [37] introduced a gating mechanism that allows agents to learn when to communicate, producing sparse communication

patterns. More recent works have focused on learning sparse communication topologies to reduce communication overhead while preserving coordination quality. SchedNet [38] proposed a weight-based scheduler for prioritizing agents' messages, and I2C [5] introduced an information-theoretic objective that maximizes mutual information between communicated messages and recipients' actions while penalizing communication frequency. It is worth noting that most learned-communication MARL works focus on completing high-level collaborative missions, while adopting relatively abstract channel models for underlying wireless transmission. Orthogonally, a separate line of research concentrates on physical-layer resource optimization for UAV networks. These works jointly optimize trajectory, transmit power, beam-forming and energy-aware data collection under realistic interference, bandwidth and energy constraints [39–41]. In addition, Ge et al. reviewed the importance of sampling and communication for coordination in multi-agent systems. In real-world UAV deployment scenarios, perception updates and wireless transmission may be asynchronous or temporarily interrupted, resulting in outdated information and a decline in cooperative performance [42].

These works have established the importance of sparse, targeted communication in multi-agent systems, but have not been integrated with GNN-based relational reasoning for heterogeneous team coordination. Our edge predictor, trained jointly with the policy and regularized by an L1 sparsity penalty, builds upon this line of research by enabling learnable, task-adaptive communication topology within a GNN-based MARL framework, specifically tailored to the heterogeneous multi-UAV air-to-ground mission planning setting. In summary, while significant progress has been made in MARL algorithms, GNN-based multi-agent architectures, and communication topology learning as separate research threads, their integration into a unified framework for heterogeneous multi-UAV air-to-ground mission planning remains an open challenge.

3. Problem Formulation

3.1. Dec-POMDP Model

We formalize the multi-UAV collaborative air-to-ground mission as a decentralized partially observable Markov decision process (Dec-POMDP) defined by the tuple:

$$M = \langle S, \{O_i\}_{i=1}^N, \{A_i\}_{i=1}^N, T, R, Z, \gamma \rangle \quad (1)$$

where N denotes the number of heterogeneous UAV agents, and the UAV types include reconnaissance, strike, and electronic warfare, denoted by the label $\text{role}_i = \{\text{RECON}, \text{STRIKE}, \text{EW}\}$. Therefore, the total number of agents is:

$$N = N_{\text{RECON}} + N_{\text{STRIKE}} + N_{\text{EW}} \quad (2)$$

S is the global state space. The global state $s \in S$ encompasses, for each UAV i , its position (x_i, y_i) , altitude h_i , fuel level f_i , ammunition count am_i , stealth mode status, and active status; for each SAM site k , its position (x_k, y_k) , active status; and for each other ground target j , its position (x_j, y_j) and active status. Formally, the global state vector is:

$$s = [x_1, y_1, h_1, a_{\text{active}}^{\text{uav}_1}, f_1, am_1, a_{\text{stealth}}^{\text{uav}_1}, \dots, x_1^{\text{SAM}}, y_1^{\text{SAM}}, a_{\text{active}}^{\text{SAM}_1}, \dots, x_1^{\text{tgt}}, y_1^{\text{tgt}}, a_{\text{active}}^{\text{tgt}_1}, \dots] \quad (3)$$

where $a_{\text{active}}^{\text{uav}_i}$ includes the heading ψ_i , speed v_i and survival status of UAV i , $a_{\text{active}}^{\text{SAM}_k}$ is the survival status of SAM k , and $a_{\text{active}}^{\text{tgt}_j}$ is the survival status of other ground target j .

O_i is the local observation space for agent i . Each UAV receives only partial information: its own kinematic state, sensor readings, and messages from communicating teammates within range. $A_i = A_i^{\text{cont}} \times A_i^{\text{disc}}$ is the mixed action space. $T: S \times A_1 \times \dots \times A_N \rightarrow \Delta(S)$ is the stochastic state transition function governed by the physical models. $R: S \times A_1 \times \dots \times A_N \rightarrow \mathbb{R}^N$ is the team-shared reward function. $Z: S \times A \times O_1 \times \dots \times O_N \rightarrow [0, 1]$ is the observation function:

$$Z(s', a, o_1, \dots, o_N) = \prod_{i=1}^N Z_i(o_i | s', a) \quad (4)$$

which defines the probability of each agent receiving observation o_i after the system transitions to state s' due to joint action a . $\gamma = 0.99$ is the discount factor.

The objective is to learn decentralized policies $\pi_i(a_i | o_i)$ for each agent that maximize the expected cumulative team reward:

$$J(\pi_1, \dots, \pi_N) = \mathbb{E}_{s_0, a_t \sim \pi, s_{t+1} \sim T} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] \quad (5)$$

3.2. Mixed Action Space

Each UAV agent i outputs a hybrid action $a_i = (a_i^{\text{cont}}, a_i^{\text{disc}})$ defined as continuous action $a_i^{\text{cont}} \in [-1, 1]$, where $a_{i,1}^{\text{cont}}$ is heading rate ψ_i (normalized to $[-1, 1]$, mapped to $[-\omega_{\max}, \omega_{\max}]$), $a_{i,2}^{\text{cont}}$ is speed change v_i (normalized to $[-1, 1]$, mapped to $[-a_{\max}, a_{\max}]$), and $a_{i,3}^{\text{cont}}$: desired altitude change h_i (normalized to $[-1, 1]$, mapped to $[-h_{\max}, h_{\max}]$).

Discrete action $a_i^{\text{disc}} = (a_i^{\text{tgt}}, a_i^{\text{def}}, a_i^{\text{comm}})$ comprises three heads: $a_i^{\text{tgt}} \in \{0, 1, \dots, M\}$ is target engagement selection, where 0 indicates “no target” and $1, \dots, M$ selects among M targets. $a_i^{\text{def}} \in \{0, 1, 2, 3\}$ is defense mode selection, from $\{\text{PASSIVE}, \text{CHAFF}, \text{JAMMER}, \text{MANEUVER}\}$ (in Table 1). $a_i^{\text{comm}} \in \{0, 1\}$ is a communication broadcast flag, indicating whether to transmit current state and detections to teammates.

Table 1. Defense mode enumeration.

Mode	Index	Description
PASSIVE	0	No active defense, relies on stealth.
CHAFF	1	Deploy chaff to confuse radar tracking.
JAMMER	2	Activate on-board jammer (EW only).
MANEUVER	3	Execute evasive maneuvering to break radar lock.

3.3. Reward Design

The reward function combines sparse event-based signals with dense shaping components to facilitate learning in an otherwise sparse-reward domain. The sparse event rewards are:

$$r_i^{\text{kill}} = \begin{cases} +20, \text{ ground target destroyed} \\ +50, \text{ SAM destroyed} \\ +15, \text{ coordinated strike (EW within 1Km of target at time of engagement)} \\ -100, \text{ UAV destroyed by SAM} \end{cases} \quad (6)$$

Dense shaping rewards consist of proximity bonus, SAM danger penalty and per-step penalty, where proximity bonus is a weak signal encouraging agents to close distance toward the nearest active target:

$$r_i^{\text{prox}} = \beta_{\text{prox}} \cdot \left(1 - \min \left(\frac{d_{\text{nearest}}}{r_{\text{sensor}}}, 2 \right) \right), \beta_{\text{prox}} = 0.001 \quad (7)$$

where d_{nearest} is the minimum Euclidean distance from UAV i to any active target, and r_{sensor} is the UAV's sensor range.

SAM danger penalty penalizes each UAV for being inside any SAM's engagement range:

$$r_i^{\text{danger}} = -\beta_{\text{danger}} \cdot \left(1 - \frac{d_{\text{closest_SAM}}}{R_{\text{engage}}^{\text{SAM}}} \right) \cdot a_{\text{active}}^{\text{SAM},k}, \beta_{\text{danger}} = 0.01 \quad (8)$$

$$a_{\text{active}}^{\text{SAM},k} = \begin{cases} 1, d_{\text{closest_SAM}} \leq R_{\text{engage}}^{\text{SAM}} \\ 0, d_{\text{closest_SAM}} > R_{\text{engage}}^{\text{SAM}} \end{cases} \quad (9)$$

where $d_{\text{closest_SAM}}$ is the distance from UAV i to the nearest SAM that has brought the UAV i within engagement range. $R_{\text{engage}}^{\text{SAM}}$ is the shoot range of SAM.

Per-step penalty is $r_{\text{step}} = -0.01$, incentivizing mission efficiency and discouraging loitering. The total per-agent reward at each timestep is:

$$r_i = r_i^{\text{kill}} + r_i^{\text{prox}} + r_i^{\text{danger}} + r_{\text{step}} \quad (10)$$

4. System Model

Figure 1 illustrates a conceptual diagram of an air-to-ground mission, in which the defending force uses SAMs to protect ground targets; the blue circle represents the SAM radar detection range and the red circle represents the SAM engagement range. The UAV swarm consists of reconnaissance, strike, and electronic-warfare UAVs. The reconnaissance UAVs are responsible for detecting SAMs and ground targets; the electronic-warfare UAVs are responsible for jamming SAM radars to ensure the safe entry of strike UAVs into the area; and the strike UAVs are responsible for destroying SAMs and ground targets. Explanations of the relevant symbols are provided in Appendix A.

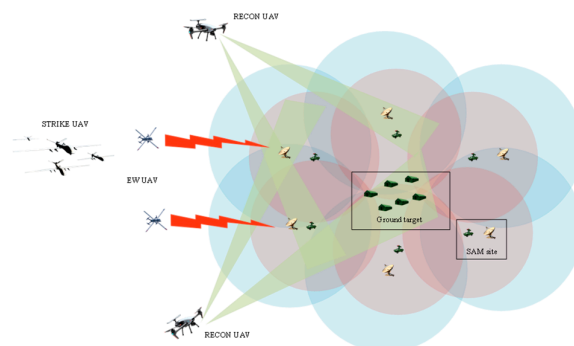


Figure 1. Conceptual diagram of an air-to-ground mission. The blue circles indicate the detection range of SAM radars, and the red circles indicate the range of the SAMs.

4.1. Battlefield Environment

The battlefield environment is defined over a field of dimensions $6000 \text{ m} \times 6000 \text{ m} \times 15000 \text{ m}$. All UAVs are generated at the origin, with positions uniformly distributed on one of the four edges. Initial heading is uniformly random in $[0, 2\pi)$. SAM sites are deployed in a ring around the center of the battlefield. For K SAM sites, the base angular position of the SAM k is:

$$\theta_k^{\text{base}} = \frac{2\pi k}{K} + \varepsilon_k, \quad \varepsilon_k \sim \mathcal{U}(-0.2, 0.2) \text{ rad} \quad (11)$$

$$P_k^{\text{SAM}} = \left(\frac{X}{2}, \frac{Y}{2} \right) + R_{\text{engage}}^{\text{SAM}} \cdot (\cos \theta_k^{\text{base}}, \sin \theta_k^{\text{base}}) \quad (12)$$

where $R_{\text{engage}}^{\text{SAM}}$ is the SAM shoot range. Ground targets are randomly placed within a radius $R_{\text{tgt}} = 2 \text{ Km}$ around the center.

4.2. UAV Kinematics Model

Each UAV is modeled as a 3-DOF point mass with constrained dynamics. The state variables include 2D position, x_i and y_i (m), heading angle ψ (rad), speed v (m/s), altitude h_i (m) and P_t is the planar position update. The discrete-time kinematic update with timestep $\Delta t = 1.0 \text{ s}$ is:

$$\psi_{t+1} = \text{wrap}(\psi_t + \delta_\psi \cdot \omega_{\text{max}} \cdot \Delta t) \quad (13)$$

$$v_{t+1} = \text{clip}(v_t + \delta_v \cdot a_{\text{max}} \cdot \Delta t, v_{\text{min}}, v_{\text{max}}) \quad (14)$$

$$h_{t+1} = \text{clip}(h_t + \delta_h \cdot v_{\text{climb}}^{\text{max}} \cdot \Delta t, h_{\text{min}}, h_{\text{max}}) \quad (15)$$

$$P_{t+1} = P_t + v_{t+1} \cdot [\cos \psi_{t+1}, \sin \psi_{t+1}]^T \cdot \Delta t \quad (16)$$

where $\delta_\psi, \delta_v, \delta_h \in [-1, 1]$ are the normalized control inputs, provided by the policy network, $\text{wrap}(\cdot)$ normalizes the angle to $[-\pi, \pi]$, and $\text{clip}(\cdot)$ enforces the physical bounds. ω_{max} is the maximum yaw angular velocity, a_{max} is the maximum horizontal acceleration, v_{min} and v_{max} are the minimum and maximum horizontal velocities, $v_{\text{climb}}^{\text{max}}$ is the maximum climb rate, and h_{min} and h_{max} are the minimum and maximum flight altitudes.

4.3. Radar Sensor Model

Each UAV is equipped with a radar for ground target detection. The on-board radar parameters are frequency $f = 10 \text{ GHz}$, wavelength $\lambda = c/f = 0.03 \text{ m}$, peak transmit power $P = 5000 \text{ W}$, antenna gain $G = 30$, and noise figure $F = 5$. The maximum detection range for a target with RCS σ is computed using the standard radar range equation:

$$R_{\text{max}}(\sigma) = \left[\frac{PG^2\lambda^2\sigma}{(4\pi)^3 kT_0 BF \cdot \text{SNR}_{\text{min}}} \right]^{1/4} \quad (17)$$

where $k = 1.38 \times 10^{-23} \text{ J/K}$ is Boltzmann's constant, $T_0 = 290 \text{ K}$ is standard noise temperature, $B = 1 \text{ MHz}$ is the receiver bandwidth, and $\text{SNR} = 13 \text{ dB}$ is the minimum detectable signal-to-noise ratio. The probability of detection at range R follows:

$$P_d(R, \sigma) = \frac{P_d^{\text{max}}}{1 + (R/R_{\text{max}}(\sigma))^4} \quad (18)$$

with $P_d^{\max} = 0.95$. When a detection occurs, the radar provides range and RCS estimates corrupted by multiplicative gaussian noise scaled by the normalized range:

$$\hat{R} = R \cdot (1 + \eta_R), \eta_R \sim N(0, 0.05 \cdot R / R_{\max}) \quad (19)$$

$$\hat{\sigma} = \sigma \cdot (1 + \eta_\sigma), \eta_\sigma \sim N(0, 0.1) \quad (20)$$

4.4. Radar Warning Receiver (RWR)

Each UAV carries a passive radar warning receiver (RWR) that detects SAM radar emissions. The RWR can detect radar signals at 1.5 times the SAM's own radar range. For each SAM k with radar range $R_{\text{radar}}^{\text{SAM}}$, the RWR on UAV i at distance d_{ik} detects emissions if:

$$d_{ik} < R_{\text{radar}}^{\text{SAM}} \cdot \eta_{\text{rwr}}, \eta_{\text{rwr}} = 1.5 \quad (21)$$

For each detected emission, the RWR reports:

- (1) Azimuth of the radiation source:

$$\phi_k = \arctan2(y_k - y_i, x_k - x_i) \quad (22)$$

- (2) Threat level on $[0, 1]$:

$$\text{threat}_{ik} = \begin{cases} 0.7 + 0.3 \cdot (1 - d_{ik} / R_{\text{detect}}), & d_{ik} < R_{\text{engage}}^{\text{SAM}} \\ 0.3 + 0.4 \cdot (1 - d_{ik} / R_{\text{detect}}), & R_{\text{engage}}^{\text{SAM}} \leq d_{ik} < R_{\text{radar}}^{\text{SAM}} \\ 0.1 + 0.2 \cdot (1 - d_{ik} / R_{\text{detect}}), & d_{ik} \geq R_{\text{radar}}^{\text{SAM}} \end{cases} \quad (23)$$

where $R_{\text{detect}} = R_{\text{radar}}^{\text{SAM}} \cdot \eta_{\text{rwr}}$ is the RWR detection range.

The aggregate risk score for UAV i combines all RWR signals:

$$\text{risk}_i = \min \left(\sum_{\text{sig}} [60 \cdot a_{\text{in_engage}} + 40 \cdot \text{threat}_{\text{sig}}], 100 \right) \quad (24)$$

where $a_{\text{in_engage}} = 1$, if $d_{ik} < R_{\text{engage}}^{\text{SAM}}$, for any signal. The score is capped at 100. A UAV within the kill range of a SAM poses a more critical and lethal threat; therefore, it is assigned a higher weight of 60. The cumulative threat index output by the radar warning receiver represents secondary threat information and is assigned a weight of 40. The $\text{risk}_i > 65$ has been manually calibrated to ensure that stealth mode is activated before the risk value reaches the upper limit of 100, thereby reserving a margin of response for other defensive measures.

4.5. SAM Site Model

SAM sites represent the primary threat to UAVs. The SAM radar detects UAV i at distance d_{ik} with current RCS σ_i using an RCS-scaled effective range:

$$R_{\text{eff}} = R_{\text{radar}}^{\text{SAM}} \cdot \sqrt[4]{\frac{\sigma_i}{\sigma_{\max}}} \quad (25)$$

where $R_{\text{radar}}^{\text{SAM}}$ is the detection range of the SAM radar when the UAV has maximum RCS $\sigma_{\max}$.

Detection is successful if $d_{ik} \leq R_{\text{eff}}$. If d_{ik} exceeds $R_{\text{radar}}^{\text{SAM}}$, detection fails. For intermediate cases ($R_{\text{eff}} < d_{ik} \leq R_{\text{radar}}^{\text{SAM}}$), there is a 10% false alarm probability.

When an EW UAV with jamming power P_{jam} is within the jamming range, the effective radar detection range is reduced by the jammer-to-signal ratio:

$$R_{\text{max}}^{\text{eff}} = R_{\text{eff}} \cdot \max\left(0.1, 1 - \frac{J/S_{\text{dB}}}{30}\right) \quad (26)$$

the floor of 0.1 ensures that even under heavy jamming, close-range detections remain possible. The jammer-to-signal ratio is computed:

$$J/S = \frac{P_{\text{jam}}}{P_t^{\text{SAM}}} \cdot \frac{R_{\text{radar_dist}}^2}{R_{\text{jam_dist}}^2} \quad (27)$$

$$J/S_{\text{dB}} = 10 \log\left(\max(J/S, 10^{-10})\right) \quad (28)$$

where $R_{\text{radar_dist}}$ is the distance from the SAM to the targeted UAV, and $R_{\text{jam_dist}}$ is the distance from the SAM to the EW UAV that is suppressing this SAM. If $J/S_{\text{dB}} > 3 \text{ dB}$, the radar is considered jammed and detection fails entirely.

Once detected, the SAM maintains a track on UAV i with quality:

$$Q_{\text{track}} = \alpha_{\text{track}} \cdot \max\left(0, 1 - \left(\frac{d_{ik}}{R_{\text{radar}}^{\text{SAM}}}\right)^2\right) \cdot \min\left(1, \frac{\sigma_i}{\sigma_{\text{max}}}\right) \quad (29)$$

Track quality decays by 0.1 per step when the target leaves radar range:

$$Q_{\text{track}} \leftarrow \max(0, Q_{\text{track}} - 0.1) \quad (30)$$

The SAM fires a missile when all of the following conditions are met:

- (1) Cooldown counter $c_k = 0$.
- (2) Distance $d_{ik} \leq R_{\text{engage}}^{\text{SAM}}$.
- (3) Track quality $Q_{\text{track}} \geq 0.3$.

The probability of a kill is:

$$P_k = P_k^{\text{base}} \cdot Q_{\text{track}} \cdot \exp\left(-\frac{d_{ik}}{2R_{\text{engage}}^{\text{SAM}}}\right) \quad (31)$$

A uniform random draw $u \sim U(0, 1)$ determines the outcome: the UAV is shot down if $u < P_k$. After firing, the SAM sets $c_k = C_{\text{max}}$. Figure 2 shows the complete SAM engage procedure.

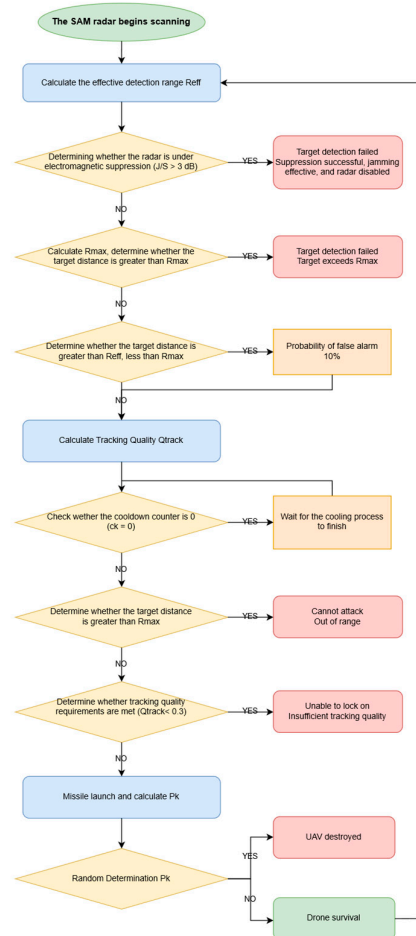


Figure 2. The complete SAM engage procedure.

4.6. Inter-UAV Communication Model

Communication between UAVs is modeled as a distance-dependent link quality graph. For any pair of active UAVs (i_1, i_2) separated by distance $d_{i_1 i_2}$:

$$Q_{i_1 i_2} = \begin{cases} 1, & i_1 = i_2 \\ 0, & d_{i_1 i_2} > R_{\text{comm}}^{\text{max}} \\ 1 / 1 + \left(d_{i_1 i_2} / d_0 \right)^2, & d_{i_1 i_2} \leq R_{\text{comm}}^{\text{max}} \end{cases} \quad (32)$$

where $R_{\text{comm}}^{\text{max}}$ is the maximum communication range and $d_0 = 0.3 \cdot R_{\text{comm}}^{\text{max}}$ is the distance at which link quality drops to 0.5.

When a UAV broadcasts, discrete action $a_i^{\text{comm}} = 1$, it transmits a message containing:

$$\text{msg}_i = \{uid, P_i, \psi_i, v_i, h_i, role_i, D_i^{\text{detect}}, am_i, f_i\} \quad (33)$$

where uid is used to identify which UAV the information comes from; it is the UAV's unique identifier. D_i^{detect} is the set of target IDs currently detected by on-board radar on UAV i . A message from sender i_1 to receiver i_2 is dropped with probability:

$$P_{\text{drop}} = \min \left(P_{\text{drop}}^{\text{base}} + 0.5 \cdot (1 - Q_{i_1 i_2}), 0.95 \right) \quad (34)$$

where $p_{\text{drop}}^{\text{base}} = 0$, and messages over low-quality links fail with up to 95% probability.

5. Proposed Method

In this section, we will provide a detailed explanation of the SAGA, including its architectural design and training process.

5.1. CTDE Architecture

The CTDE architecture separates the learning paradigm into two phases. During centralized training, a critic network $V_{\phi}(s)$ observes the complete global state s , providing accurate value estimates that account for the full multi-agent state. During decentralized execution, each UAV i relies solely on its local observation o_i through a shared-parameter actor $\pi_{\theta}(a_i | o_i)$. Actor parameters θ are shared across all agents regardless of role. The critic is discarded at execution time, and each UAV acts purely on local information. The architecture is illustrated in Figure 3.

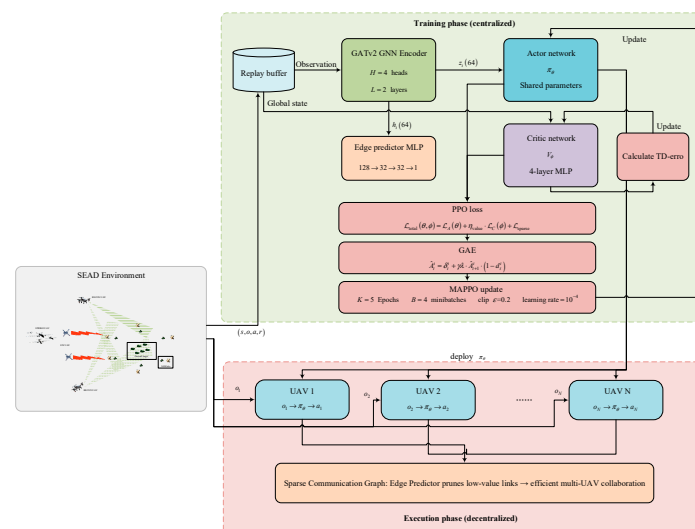


Figure 3. SAGA's CTDE architecture.

At training time, it is possible to access observations from all UAVs. However, during decentralized execution, there exists no central entity that collects raw observations from every UAV. Each UAV only has access to its own local observation. The node embeddings of neighboring teammates are obtained by receiving communication messages. Each UAV encodes its local observation into a node embedding vector, which is transmitted to teammates when $a_i^{\text{comm}} = 1$. Each UAV then performs local GATv2 attention and edge predictor computation using its self-produced embedding together with embeddings received from communication neighbors.

5.2. GNN-Enhanced Actor Network

To enable implicit coordination through learned graph reasoning, we employ a graph neural network encoder built on GATv2 convolutions. The observation-to-latent pipeline proceeds in three stages: observation encoding, GATv2 message passing, and learned edge prediction. The complete actor network architecture is illustrated in Figure 4.

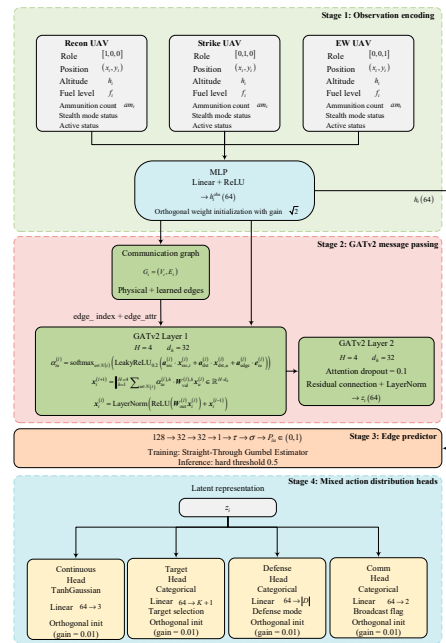


Figure 4. SAGA's actor network architecture.

5.2.1. Observation Encoding

Each UAV's raw observation o_i is independently projected to an initial node feature vector through the MLP:

$$\mathbf{x}_i^{(0)} = \text{MLP}_{\text{obs}}(o_i) \in \mathbb{R}^{64} \quad (35)$$

where all layers use ReLU activation and orthogonal weight initialization with gain $\sqrt{2}$. This encoding maps each UAV's heterogeneous sensor and communication data into a compact representation suitable for graph-based reasoning.

5.2.2. GATv2 Message Passing

We use the GATv2 convolution, which computes attention as a joint nonlinear function of source node, destination node, and edge features, thereby avoiding the static attention limitation of the original GAT. Two message-passing layers are stacked, each corresponding to one round of distributed consensus, and operates on the communication graph $G = (V, E)$.

For each GNN layer, $l \in \{1, 2\}$ with $H = 4$ attention heads and per-head dimension $d_h = 64 / H = 16$. The source and destination projections are:

$$\mathbf{x}_{\text{src}}^{(l)} = \mathbf{W}_{\text{src}}^{(l)} \mathbf{x}_{\text{src}}^{(l-1)} \in \mathbb{R}^{N \times H \times d_h}, \quad \mathbf{x}_{\text{dst}}^{(l)} = \mathbf{W}_{\text{dst}}^{(l)} \mathbf{x}_{\text{dst}}^{(l-1)} \in \mathbb{R}^{N \times H \times d_h} \quad (36)$$

where $\mathbf{W}_{\text{src}}^{(l)}$ and $\mathbf{W}_{\text{dst}}^{(l)}$ are the learnable projection weight matrices for the source and destination nodes. Edge feature projection is:

$$\mathbf{e}_{i_1 i_2}^{(l)} = \mathbf{W}_{\text{edge}}^{(l)} \mathbf{e}_{i_1 i_2} \in \mathbb{R}^{H \times d_h} \quad (37)$$

The GATv2 attention score for edge $i_1 \rightarrow i_2$ is calculated jointly using the concatenated source, destination, and edge features:

$$\alpha_{i_1 i_2}^{(l)} = \text{softmax}_{i_2 \in N(i_1)} \left(\text{LeakyReLU}_{0.2} \left(\mathbf{a}_{\text{src}}^{(l)} \cdot \mathbf{x}_{\text{src}, i_1}^{(l)} + \mathbf{a}_{\text{dst}}^{(l)} \cdot \mathbf{x}_{\text{dst}, i_2}^{(l)} + \mathbf{a}_{\text{edge}}^{(l)} \cdot \mathbf{e}_{i_1 i_2}^{(l)} \right) \right) \quad (38)$$

where $\mathbf{a}_{\text{src}}^{(l)}, \mathbf{a}_{\text{dst}}^{(l)}, \mathbf{a}_{\text{edge}}^{(l)} \in \mathbb{R}^{H \times d_h}$ are learnable attention vectors representing the attention scores for the source node features, destination node features, and edge features. Additionally, the sum is over the last dimension d_h after the element-wise product, yielding per-head scalar scores. The softmax is applied per destination node over all incoming edges.

Node features are updated via multi-head aggregation with residual connection and LayerNorm:

$$\mathbf{x}_{i_1}^{(l+1)} = \left\| \sum_{h=1}^{H=4} \sum_{i_2 \in N(i_1)} \alpha_{i_1 i_2}^{(l),h} \cdot \mathbf{W}_{\text{val}}^{(l),h} \mathbf{x}_{i_2}^{(l)} \right\| \in \mathbb{R}^{H \times d_h} \quad (39)$$

where $\|$ denotes concatenation across H heads, $\mathbf{W}_{\text{val}}^{(l),h}$ is the learnable weight matrix for the value matrix, and $\mathbf{x}_{i_2}^{(l)}$ is the input feature vector of node i_2 in layer l . Each GNN layer thus corresponds to one round of distributed consensus: nodes aggregate information from communication neighbors weighted by learned attention, update their representations, and normalize through LayerNorm. Output projection with residual connection is:

$$\mathbf{x}_{i_1}^{(l)} = \text{LayerNorm} \left(\text{ReLU} \left(\mathbf{W}_{\text{out}}^{(l)} \mathbf{x}_{i_1}^{(l+1)} \right) + \mathbf{x}_{i_1}^{(l-1)} \right) \quad (40)$$

where $\mathbf{W}_{\text{out}}^{(l)}$ is the learnable linear projection weight matrix for the output branch, nodes aggregate information from their communication neighbors, where the weight of this information is determined by learned attention parameters, update their representations, and pass them forward.

5.2.3. Edge Predictor

While the physical communication graph provides the base connectivity structure, not all physically possible links are equally valuable for coordination. An auxiliary edge predictor module learns which UAV pairs should actively communicate, enabling sparse and efficient message passing. The predicted score is:

$$\text{Log}_{i_1 i_2} = \text{MLP}_{\text{edge}} \left(\mathbf{x}_{i_1}^{(0)} \parallel \mathbf{x}_{i_2}^{(0)} \right) \quad (41)$$

where MLP_{edge} has structure $128 \rightarrow 32 \rightarrow 32 \rightarrow 1$ with ReLU activations. The probability of an edge existing is:

$$p_{i_1 i_2}^{\text{edge}} = \sigma \left(\frac{\text{Log}_{i_1 i_2}}{\tau} \right) \quad (42)$$

where τ is a temperature parameter and σ is the sigmoid function. Self-loops are fixed at $p_{i_1 i_2}^{\text{edge}} = 1.0$ to ensure self-updating. During training, edges are sampled via straight-through estimation, a small noise $\epsilon \sim \text{U}(0, 0.1)$ is added, and edges are binarized at threshold 0.5, but gradients flow through the continuous probabilities. During inference, hard thresholding at 0.5 is used. The predicted topology filters the physical communication graph, enabling the GNN to route messages through the most informative links.

The edge predictor is trained with an L1 sparsity penalty to incentivize communication efficiency:

$$\mathcal{L}_{\text{sparse}} = \frac{1}{N(N-1)} \sum_{i_1=1}^N \sum_{i_2 \neq i_1}^N p_{i_1 i_2}^{\text{edge}} \quad (43)$$

The edge predictor generates a logical intended communication topology, which selects which UAV pairs are supposed to exchange tactical information. This logical intention cannot guarantee successful physical delivery. After logical edge selection, actual

message transmission is still subject to physical link-quality rules in Equation (34). Even for an active predicted logical edge (i, j) , the transmitted message can still be dropped with probability P_{drop} . If physical dropping occurs, no information is received by UAV j , and this edge contributes nothing to GATv2 message passing in the current timestep. The learned sparse topology only reduces the set of intended communication partners; it cannot eliminate physical-layer packet loss.

In the current implementation, physical edges from the communication graph serve as the base topology, and the learned edges provide augmentation. The edge predictor produces $N \times N$ edge masks that are merged with the physical graph structure. The architecture of the edge predictor is shown in Figure 5.

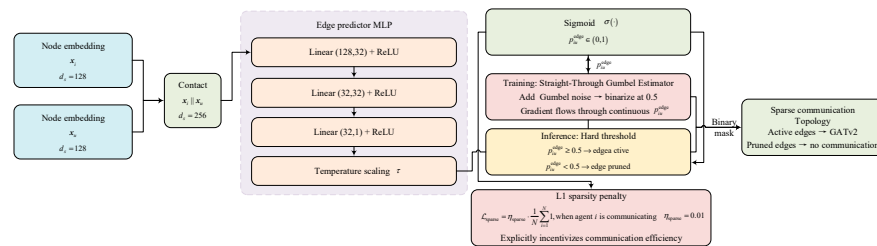


Figure 5. Architecture of the edge predictor.

5.3. Mixed Action Distribution and Output Heads

From the latent representation $\mathbf{x}_i^{(l)} \in \mathbb{R}^{64}$, four output heads produce a mixed continuous–discrete action distribution. All linear layers in the action heads use orthogonal initialization with a gain of 0.01 and zero biases, ensuring small initial action magnitudes for cautious exploration.

The continuous head uses a Tanh-squashed Gaussian distribution with state-dependent mean and globally shared learnable log-standard-deviation. The mean vector of the Gaussian distribution of the consecutive actions of the UAV i is:

$$\boldsymbol{\mu}_i = \mathbf{W}_\mu \mathbf{x}_i^{(l)} + \mathbf{b}_\mu, \mathbf{W}_\mu \in \mathbb{R}^{3 \times 64} \quad (44)$$

$$\log \sigma \in \mathbb{R}^3 \quad (45)$$

where $\mathbf{W}_\mu$ is the learnable weight matrix that generates the mean, with log-standard-deviation $\log \sigma = 0$ and constraints $\log \sigma \in [-20, 2]$, providing broad initial exploration that contracts as training progresses. The continuous action is sampled via reparameterization and Tanh squashing:

$$\mathbf{u}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \text{diag}(\sigma^2)), \mathbf{a}_i^{\text{cont}} = \tanh(\mathbf{u}_i) \quad (46)$$

The log-probability includes the change-of-variables correction term:

$$\log \pi(\mathbf{a}_i | \mathbf{o}_i) = \sum_{c=1}^3 \left[\log \mathcal{N}(u_{i,c} | \mu_{i,c}, \sigma_c^2) - \log(1 - \tanh^2(u_{i,c})) \right] \quad (47)$$

where, for numerical stability, the correction is implemented as:

$$\log \pi(\mathbf{a}_i | \mathbf{o}_i) = \log \mathcal{N}(\mathbf{u}_i | \boldsymbol{\mu}_i, \sigma^2) - \sum_{c=1}^3 (2 \log 2 - 2u_{i,c} - 2 \cdot \text{softplus}(-2u_{i,c})) \quad (48)$$

Discrete heads are represented by three independent categorical distributions, one per discrete action head:

$$\pi(a_i^{\text{tgt}} = g_{\text{tgt}} | \mathbf{o}_i) = \text{softmax}(\mathbf{W}_{\text{tgt}} \mathbf{x}_i + \mathbf{b}_{\text{tgt}}), g_{\text{tgt}} \in \{0, 1, \dots, M\} \quad (49)$$

$$\pi(a_i^{\text{def}} = g_{\text{def}} | o_i) = \text{softmax}(W_{\text{def}}x_i + b_{\text{def}}), g_{\text{def}} \in \{0, 1, 2, 3\} \quad (50)$$

$$\pi(a_i^{\text{comm}} = g_{\text{comm}} | o_i) = \text{softmax}(W_{\text{comm}}x_i + b_{\text{comm}}), g_{\text{comm}} \in \{0, 1\} \quad (51)$$

Total log-probability and entropy: The mixed distribution's log-probability and entropy are the sum of continuous and discrete components:

$$\log \pi(a_i | x_i^{(l)}) = \log \pi_{\text{cont}}(a_i^{\text{cont}} | x_i^{(l)}) + \log \pi_i^{\text{tgt}}(g_{\text{tgt}}) + \log \pi_i^{\text{def}}(g_{\text{def}}) + \log \pi_i^{\text{comm}}(g_{\text{comm}}) \quad (52)$$

$$H(\pi_i) = H_{\text{cont}}(\pi_i) + H_{\text{tgt}}(\pi_i) + H_{\text{def}}(\pi_i) + H_{\text{comm}}(\pi_i) \quad (53)$$

where H_{cont} approximates the Tanh-squashed Gaussian entropy with the base Gaussian entropy:

$$H_{\text{cont}} \approx \sum_{c=1}^3 \frac{1}{2} (1 + \log 2\pi\sigma_c^2) = \frac{3}{2} (1 + \log 2\pi) + \sum_{c=1}^3 \log \sigma_c \quad (54)$$

When initializing the weights, all action head linear layers use orthogonal initialization with gain of 0.01 and zero biases. The small gain ensures initial near-zero action means, promoting cautious exploration. The continuous log-std parameter is initialized at 0, providing broad initial exploration that contracts as training progresses.

5.4. Centralized Critic Network

In the CTDE paradigm, the critic network provides a centralized value function V_ϕ that estimates the expected return from the global states. By conditioning on the full multi-agent state, the critic overcomes the partial observability that individual agents face during execution, yielding accurate value estimates for advantage computation. The critic is used exclusively during training; at execution time it is discarded entirely. The complete critic network architecture is illustrated in Figure 6.

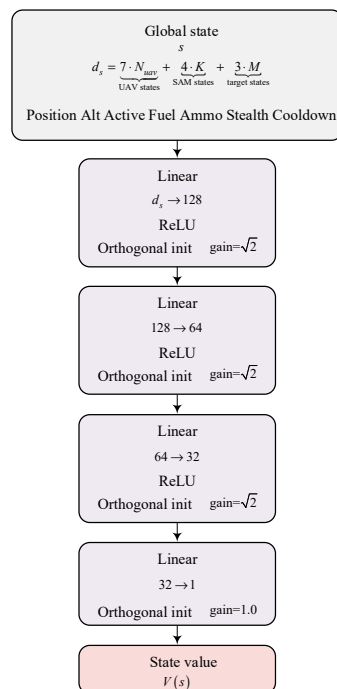


Figure 6. SAGA's critic network architecture.

The critic network estimates the state value from the global state. The architecture is a 4-layer MLP with monotonically decreasing widths:

$$\mathbf{h}^{(1)} = \text{ReLU}\left(\mathbf{W}^{(1)}\mathbf{s} + \mathbf{b}^{(1)}\right), d_s \rightarrow 128 \quad (55)$$

$$\mathbf{h}^{(2)} = \text{ReLU}\left(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\right), 128 \rightarrow 128 \quad (56)$$

$$\mathbf{h}^{(3)} = \text{ReLU}\left(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}\right), 128 \rightarrow 64 \quad (57)$$

$$V_\phi(\mathbf{s}) = \mathbf{W}^{(4)}\mathbf{h}^{(3)} + \mathbf{b}^{(4)}, 64 \rightarrow 1 \quad (58)$$

Orthogonal initialization with gain $\sqrt{2}$ is used for all layers except the final layer (gain 1.0), and all biases are initialized to zero. The critic is only used during centralized training and is not required for decentralized execution. The global state dimension d_s depends on the scenario configuration:

$$d_s = \underbrace{7 \cdot N}_{\text{UAV states}} + \underbrace{3 \cdot K}_{\text{SAM states}} + \underbrace{3 \cdot M}_{\text{target states}} \quad (59)$$

where each UAV contributes 7 features, each SAM contributes 3 features, and each target contributes 3 features.

5.5. Training Algorithm

The training of the proposed MARL framework follows the centralized training with decentralized execution (CTDE) paradigm. During centralized training, a global critic network has access to the full multi-agent state and guides the policy improvement of all UAV agents, while each agent's actor network conditions only on its local observation—preserving full decentralization at execution time. The training procedure is built upon the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm and alternates between two phases: rollout collection using the current stochastic policy, and policy improvement via PPO clipped surrogate updates on the collected trajectories. The complete procedure is formalized in Algorithm 1. A theoretical analysis of algorithm time complexity is provided in Appendix B.

Algorithm 1: SAGA Algorithm

```

1: Initialize actor  $\pi_\theta$ , critic  $V_\phi(\mathbf{s})$  with orthogonal init; rollout buffer
    $B \leftarrow \emptyset$  (capacity  $T$ )
2: for iteration = 1, 2, ... do
3:   for  $t=0$  to  $T-1$  do
4:     Sample global state  $\mathbf{s}_t$ ; update graph  $G_t$ , extract edge indices  $E_t$ 
5:     Sample  $\mathbf{a}_t \sim \pi_\theta(\cdot | \mathbf{o}_t, G_t)$  via forward_graph()
6:     Execute  $\mathbf{a}_t$ ; receive  $r_t, \mathbf{o}_{t+1}, \mathbf{d}_t$ 
7:     Store  $(\mathbf{s}_t, E_t, \mathbf{o}_t, \mathbf{a}_t, \log \pi_\theta, r_t, \mathbf{d}_t)$  in  $B$ 
8:   end for
9:   Compute  $V$  via batched forward pass
10:  for  $i=1$  to  $N$  do
11:    Compute  $\delta_t^i, \hat{A}_t^i$  via GAE;  $G_t^i = \hat{A}_t^i + V_\phi(\mathbf{s}_t)$ 
12:  end for
13:   $\theta_{\text{old}} \leftarrow \theta$ 
14:  for epoch = 1 to  $K$  do
15:    Shuffle  $T$  timesteps; partition into  $B$  minibatches (graph preserved per
    timestep)
16:    for each minibatch do
17:      Reconstruct  $E_t$  from buffer;  $\hat{A}_{\text{norm}} = \hat{A} - \mu_{\hat{A}} / \sigma_{\hat{A}} + 10^{-8}$ 

```

```

18:      Compute  $\mathcal{L}_A(\theta)$ ,  $\mathcal{L}_C()$ ,  $\mathcal{L}_{\text{sparse}}$ 
19:       $\mathcal{L}_{\text{total}}(\theta, \phi) = \mathcal{L}_A(\theta) + \eta_{\text{value}} \cdot \mathcal{L}_C(\phi) + \mathcal{L}_{\text{sparse}}$ 
20:      Update  $\theta$ ,  $\phi$  via Adam; clip gradients to max norm 0.5
21:  end for
22: end for
23: end for

```

5.5.1. Objective and Rollout Collection

Under the CTDE framework, the objective is to maximize the expected cumulative discounted return over an episode of length T :

$$J(\theta) = \mathbb{E}_{\xi \sim \pi_{\theta}} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \quad (60)$$

where $\xi = (s_0, o_0, a_0, r_0, \dots, s_{T-1}, o_{T-1}, a_{T-1}, r_{T-1})$ denotes the trajectory sampled under the joint policy $\pi_{\theta} = \pi_{\theta}^1, \dots, \pi_{\theta}^N$ for N UAV agents, s_t is the global state, $o_t = o_t^1, \dots, o_t^N$ is the joint observation, $a_t = a_t^1, \dots, a_t^N$ is the joint action, r_t is the shared team reward, and γ is the discount factor.

At each training iteration, a rollout of T consecutive timesteps is collected using the current policy π_{θ} . For each timestep $t = 0, 1, \dots, T-1$, the following steps are executed:

1. Global state capture: The global state s_t is sampled from the environment before actions are taken, ensuring that s_t is the state upon which the subsequent actions are conditioned.
2. Communication graph update. The inter-UAV communication graph $G_t = (V_t, E_t)$ is refreshed based on current UAV positions and active communication masks. The edge indices and edge attributes at timestep t are extracted and stored for subsequent GNN-based processing during PPO updates.
3. Action sampling. Actions are sampled from the stochastic policy. All N agents' observations are stacked into a batch tensor and processed jointly through the graph-based actor forward pass with the current graph structure G_t , producing correlated actions for all agents in a single forward pass.
4. Environment step. The joint action a_t is executed in the environment, yielding the next joint observation o_{t+1} , the shared reward r_t , and per-agent termination flags d_t .
5. Buffer storage. All time steps t , the global state s_t , joint observation o_t , joint action a_t , action log-probabilities, reward r_t , termination flags d_t , as well as graph edge indices and attributes, are stored in a pre-allocated rollout buffer of capacity T .

5.5.2. Generalized Advantage Estimation (GAE)

Per-agent advantages are computed using Generalized Advantage Estimation (GAE) in reverse chronological order. For each agent i and each timestep $t \in 0, \dots, T-1$, the temporal-difference (TD) error is defined as:

$$\delta_t^i = r_t + \gamma \cdot V_{\phi}(s_{t+1}) \cdot (1 - d_t^i) - V_{\phi}(s_t) \quad (61)$$

where d_t^i is the termination flag for agent i at timestep t , and the term $1 - d_t^i$ ensures that the bootstrap value from the subsequent state is excluded when the episode has terminated.

The GAE advantage is computed by accumulating TD errors with exponential decay controlled by the GAE parameter:

$$\hat{A}_t^i = \delta_t^i + \gamma \lambda \cdot \hat{A}_{t+1}^i \cdot (1 - d_t^i) \quad (62)$$

The value target for each agent–timestep sample is then constructed by adding the baseline value back to the advantage:

$$G_t^i = \hat{A}_t^i + V_\phi(s_t) \quad (63)$$

This decomposition separates the training signals, the advantage $\hat{A}_t^i$, which is used for policy optimization to measure how well an action performs relative to expected returns, and the return G_t^i , which serves as the regression target for the critic network.

5.5.3. PPO Policy and Value Optimization

The collected rollout of $N \times T$ agent–timestep samples is used for K epochs of PPO updates, with B minibatches per epoch.

The policy π_θ is updated by minimizing the clipped surrogate objective, following the PPO formulation. Let $\rho_t^i(\theta)$ denote the importance sampling ratio between the current and old policies for agent i at timestep t :

$$\rho_t^i(\theta) = \exp(\log \pi_\theta(a_t^i | o_t^i) - \log \pi_{\theta_{\text{old}}}(a_t^i | o_t^i)) \quad (64)$$

To stabilize training across episodes with varying reward magnitudes, the per-agent advantages are z-score normalized within each minibatch:

$$\hat{A}_{\text{norm}} = \frac{\hat{A} - \mu_{\hat{A}}}{\sigma_{\hat{A}} + 10^{-8}} \quad (65)$$

where $\mu_{\hat{A}}$ and $\sigma_{\hat{A}}$ are, respectively, the mean and standard deviation of the advantages within the current minibatch, and the small constant 10^{-8} prevents division by zero. The actor loss is then defined as:

$$\mathcal{L}_A(\theta) = -\mathbb{E} \left[\min \left(\rho_t^i(\theta) \cdot \hat{A}_{\text{norm}}, \text{clip}(\rho_t^i(\theta), 1 - \varepsilon, 1 + \varepsilon) \cdot \hat{A}_{\text{norm}} \right) - \eta_{\text{ent}} \cdot \mathbb{E} \left[H(\pi_\theta(\cdot | o_t^i)) \right] \right] \quad (66)$$

where ε is the PPO clipping parameter that limits the magnitude of policy updates by constraining the importance sampling ratio to the interval $[1 - \varepsilon, 1 + \varepsilon]$, η_{ent} is the entropy bonus coefficient that encourages exploration by penalizing over-confident policies, and $H(\pi_\theta(\cdot | o_t^i))$ denotes the policy entropy computed analytically from the Gaussian distribution.

The centralized value function V_ϕ is trained with a clipped value loss applied to z-score normalized returns. For a minibatch of size M containing returns $\{G_1, \dots, G_M\}$, the normalization is:

$$\tilde{G} = \frac{G - \mu_G}{\sigma_G} \quad (67)$$

$$\tilde{V} = \frac{V_\phi(s) - \mu_G}{\sigma_G} \quad (68)$$

$$\tilde{V}_{\text{old}} = \frac{V_{\phi_{\text{old}}}(s) - \mu_G}{\sigma_G} \quad (69)$$

where μ_G and σ_G are the mean and standard deviation of the returns in the current minibatch. The same normalization transform is applied to the predicted values $V_\phi(s)$

and the old values $V_{\phi_{\text{old}}}(s)$, ensuring that the clipping operation between $\tilde{V}$ and $\tilde{V}_{\text{old}}$ are applied to the deviation of the normalized value relative to the clip $V_{\phi}(s)$ bound ε . The critic loss is:

$$\mathcal{L}_C(\phi) = \frac{1}{2} \mathbb{E} \left[\max \left((\tilde{V} - \tilde{G})^2, (\text{clip}(\tilde{V}, \tilde{V}_{\text{old}} - \varepsilon, \tilde{V}_{\text{old}} + \varepsilon) - \tilde{G})^2 \right) \right] \quad (70)$$

Both actor and critic parameters are updated using the Adam optimizer with learning rate $\eta = 1 \times 10^{-4}$. Gradients are clipped to a maximum L_2 -norm of 0.5 to prevent destabilizing updates from outlier samples. The approximate KL divergence is:

$$D_{\text{KL}} \approx \mathbb{E}[\rho - 1 - \log \rho] \quad (71)$$

The total gradient update for the GNN mode combines the actor loss, critic loss, and sparsity penalty:

$$\mathcal{L}_{\text{total}}(\theta, \phi) = \mathcal{L}_A(\theta) + \eta_{\text{value}} \cdot \mathcal{L}_C(\phi) + \mathcal{L}_{\text{sparse}} \quad (72)$$

where $\eta_{\text{value}} = 0.5$ is the value loss coefficient that balances the relative contribution of policy and value optimization. $\mathcal{L}_{\text{sparse}}$ is the L1 sparsity penalty in Equation (43).

6. Experiments

6.1. Experimental Environment and Parameter Configuration

To validate the effectiveness of the proposed method, a reinforcement learning training simulation environment for multi-UAV air-to-ground mission planning was built using PyCharm 2024.3.3. The deep learning framework and version used were torch 2.7.1. The simulation scenarios, weapon models, and parameters used were all virtually constructed and do not reflect any real-world conditions. The hardware configuration consisted of a Windows 10 system equipped with an Intel® Core™ i7-14700 CPU (2.10 GHz), 32 GB of RAM, and an NVIDIA RTX 4060 graphics card. The model parameters and training hyperparameters are shown in Tables 2 and 3.

Table 2. Model parameter configuration. (—) indicates that the model does not have this parameter.

Parameter	RECON UAV	STRIKE UAV	EW UAV	SAM	Ground Target
v_{max}	76.2 m/s	76.2 m/s	76.2 m/s	—	—
v_{min}	12.2 m/s	12.2 m/s	12.2 m/s	—	—
h_{max}	4572 m	4572 m	4572 m	—	—
h_{min}	152 m	152 m	152 m	—	—
ω_{max}	0.5236 rad/s	0.5236 rad/s	0.5236 rad/s	—	—
$v_{\text{climb}}^{\text{max}}$	15.4 m/s	15.4 m/s	15.4 m/s	—	—
a_{max}	15.2 m/s ²	15.2 m/s ²	15.2 m/s ²	—	—
σ_i	0.05 m ²	0.05 m ²	0.05 m ²	10 m ²	3 m ²
σ_{max}	2 m ²	2 m ²	2 m ²	10 m ²	3 m ²
f_i	100	100	100	—	—
$R_{\text{comm}}^{\text{max}}$	2000 m	2000 m	2000 m	—	—
$R_{\text{radar}}^{\text{SAM}}$	—	—	—	1500 m	—
$R_{\text{engage}}^{\text{SAM}}$	—	—	—	1200 m	—
am_i	—	5	—	—	—
	—	—	—	—	—

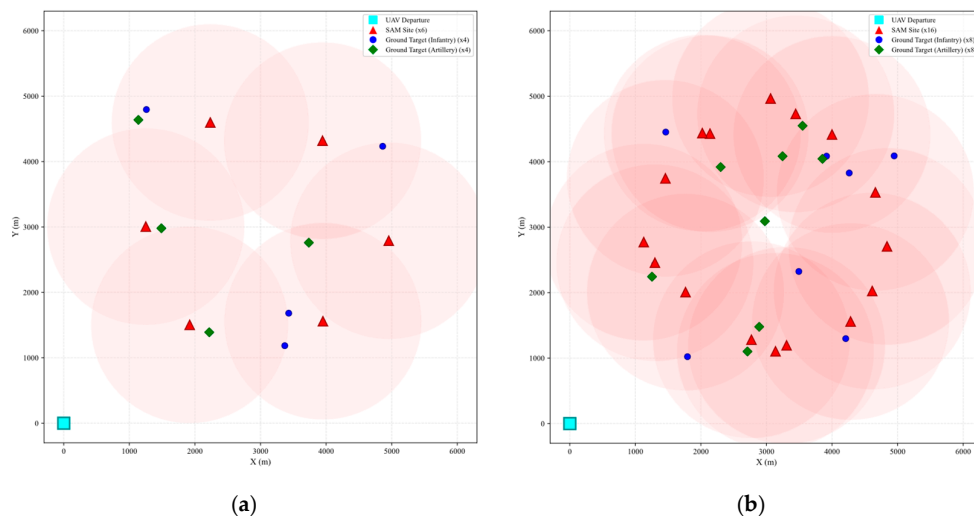
P_{jam}	—	—	150 w	—	—
$P_{\text{t}}^{\text{SAM}}$	—	—	—	5 kw	—
P_k^{base}	—	—	—	0.7	—

Table 3. Training hyperparameter configuration.

Hyperparameter	Values
Learning rate	1×10^{-4}
Steps per episode	120
Batch size	64
Discount factor	0.99
Clip range	0.2
Entropy Regularization Coefficient	0.01
GAE parameter	0.95

6.2. Comparison of Algorithm Performance

To comprehensively evaluate the performance of the algorithm proposed in this paper, we compared it with the multi-agent reinforcement learning baseline algorithms MADDPG, MATD3, QMIX, MAPPO, TarMAC, DGN and G2ANet. It should be noted that we discretize continuous maneuver dimensions into finite action bins as a standard wrapper, while discrete tactical outputs remain unchanged. This pre-processing is required to fit QMIX's monotonic value-mixing network. To ensure a fair comparison, all experiments were conducted under identical training and testing conditions. We constructed a typical mission scenario. To verify the adaptability of our method across scenarios of different scales, we specifically designed both small-scale and large-scale scenarios, with the entities and their configurations shown in Table 4 and Figure 7.

**Figure 7.** Schematic diagrams of small-scale and large-scale scenarios. (a) Small-scale. (b) Large-scale. The red circles indicate the range of the SAMs.**Table 4.** Configuration of the number of entities.

Scenarios	RECON UAV	STRIKE UAV	EW UAV	SAM	Ground Target
Small-scale	2	6	2	6	8
Large-scale	4	12	3	16	16

In terms of algorithm performance, Figure 8 shows the reward convergence curves for the four algorithms after 200,000 training iterations. The shaded area represents the standard deviation. It can be observed that SAGA exhibits superior reward convergence performance compared to the other algorithms: In small-scale scenarios, SAGA's reward rises rapidly after approximately 20,000 steps, peaks at around 390 around 70,000 steps, then enters a stable convergence phase after 80,000 steps, ultimately achieving an average reward of approximately 395. TarMAC performed slightly worse than SAGA, with a final reward value of approximately 320. MADDPG, MAPPO, DGN, and G2ANet performed similarly, with final reward values ranging from 180 to 220 and showing poor stability. QMIX performed the worst, fluctuating consistently within the 110–180 range and failing to learn an effective policy. In the large-scale scenario, SAGA surpassed 1000 after 25,000 steps and eventually stabilized in the 1080–1120 range, demonstrating clear convergence. MAPPO and TarMAC perform similarly, with a reward value of approximately 990. The reward values for all other algorithms are below 900.

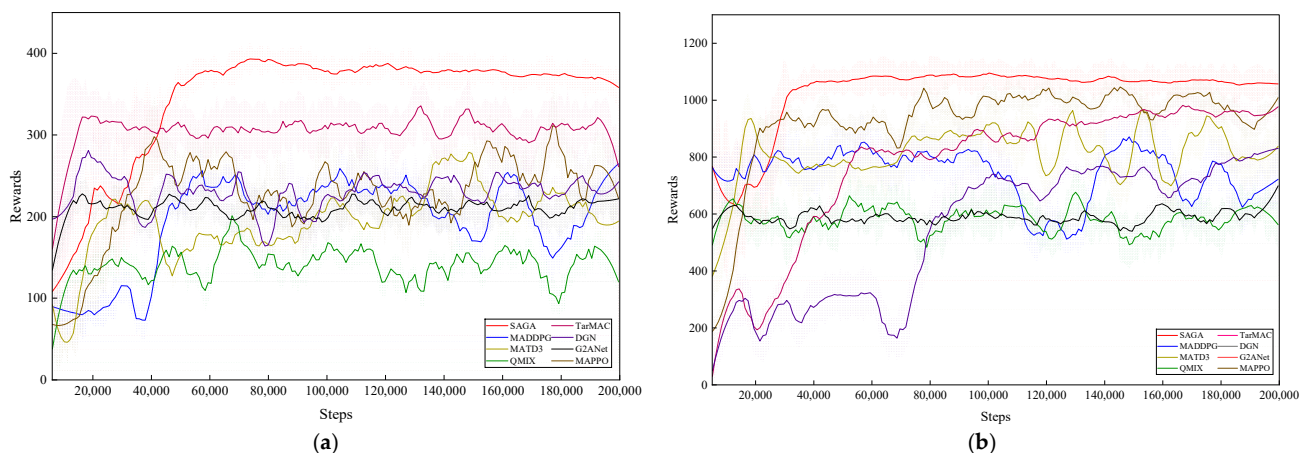


Figure 8. Comparison of algorithm training reward curves in small-scale and large-scale scenarios. (a) Small-scale. (b) Large-scale.

SAGA demonstrated the fastest convergence rate and the highest final reward in both scales, primarily attributable to the GATv2 graph attention mechanism, which enables multi-round distributed consensus via a communication graph, allowing heterogeneous agents to coordinate efficiently. The learnable edge predictor further filters out high-value communication links, avoiding information redundancy. Figure 9 shows a snapshot from a simulation, illustrating the UAVs' reconnaissance, strike, and jamming operations against the targets.

To evaluate the operational effectiveness of these algorithms, we incorporated metrics such as the average number of targets lost, average number of UAVs lost, real-time risk, and a Gantt chart showing the status of SAMs. We conducted statistical comparisons based on 100 rounds of testing, and the results are as follows.

Figure 10 shows the average number of targets lost as seen from the perspective of the UAVs. SAGA significantly outperformed other algorithms in both small- and large-scale scenarios. This indicates that the strategy learned by SAGA is not only superior in terms of reward but also far more effective in actual performance than the comparison algorithms. It is worth noting that SAGA's error bars are noticeably smaller than those of other algorithms, particularly in large-scale scenarios, indicating high policy robustness and stable performance across 100 test rounds. This demonstrates that SAGA can effectively balance the two sub-missions of suppressing air defenses and striking targets, showcasing strong coordinated capabilities.

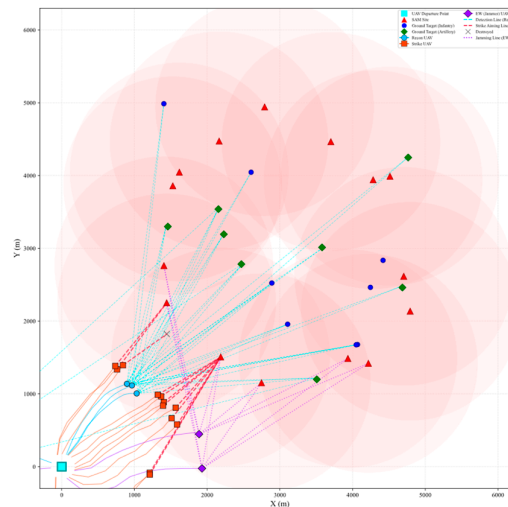


Figure 9. Snapshot from a simulation. The red circles indicate the range of the SAMs.

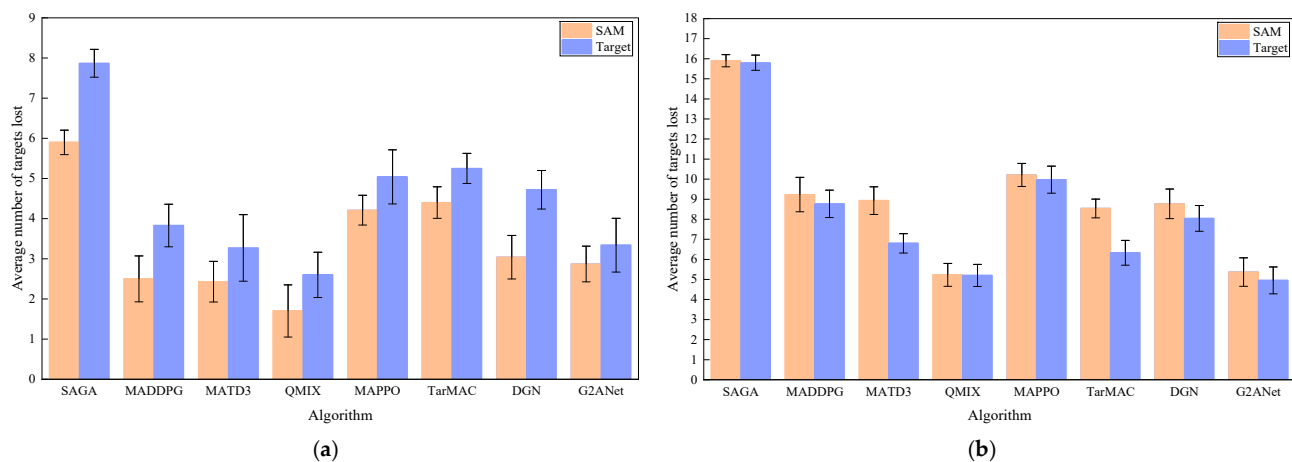


Figure 10. The average number of targets lost in small-scale and large-scale scenarios (error bars represent the standard deviation). (a) Small-scale. (b) Large-scale.

Figure 11 shows the average number of UAVs lost. In small-scale scenarios, the results were as follows: SAGA, about 1.2; MADDPG, about 4.6; MATD3, about 5.2; QMIX, about 5.9; MAPPO, about 3.7; TarMAC, about 2.7; DGN, about 5.5; and G2ANet, about 6.1. In large-scale scenarios, SAGA lost approximately 4.6 UAVs, MADDPG approximately 8.9, MATD3 approximately 9.9, QMIX approximately 11.2, MAPPO about 6.2, TarMAC about 7.1, DGN about 10.9, and G2ANet about 8.7. The reasons are as follows: (1) the GNN communication graph enables UAVs to share SAM threat information, allowing strike UAVs to proactively avoid high-risk areas; and (2) the strategies learned by EW drones and reconnaissance UAVs can indirectly provide strike UAVs with better intelligence support and a safer battlefield situation, enabling the entire formation to detect threats earlier and respond more quickly, thereby reducing losses.

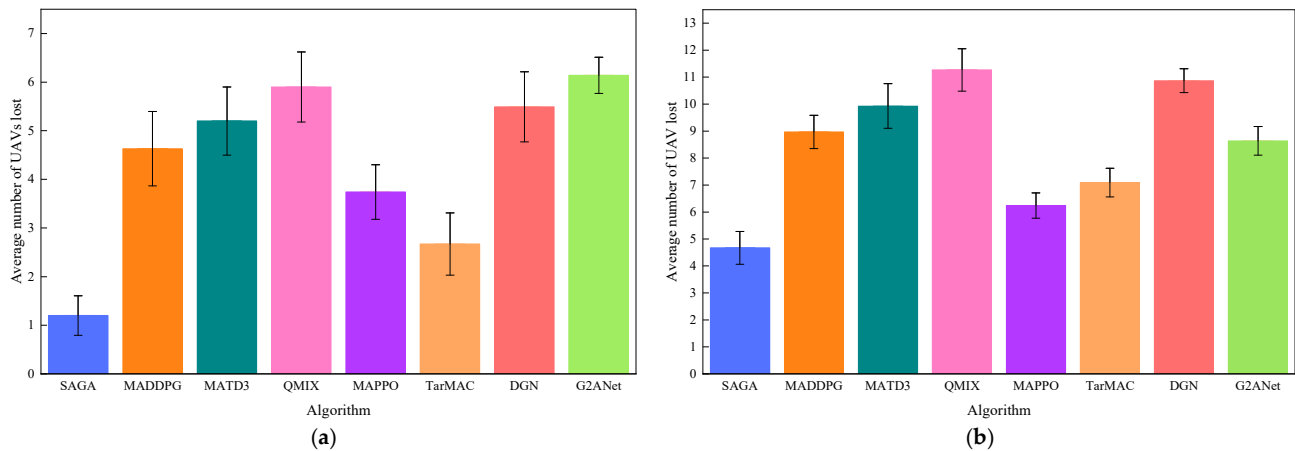


Figure 11. The average number of UAVs lost in small-scale and large-scale scenarios (error bars represent the standard deviation). (a) Small-scale. (b) Large-scale.

Figure 12a illustrates the real-time risk changes for UAV0–UAV9 during a single inference run in a small-scale scenario. Within the first 20 steps, the risks for multiple UAVs rapidly surged to nearly 100. This was the result of the UAVs entering the detection range of SAM radars and the missile’s engagement range during the penetration phase. Between steps 20 and 50, the risks fluctuate dramatically; however, each UAV is able to adopt an appropriate defensive mode to evade threats whenever it faces high risks, resulting in a short duration of high-risk periods. After step 70, the UAVs gradually exit the engagement zone, and the overall risk decreases. Figure 12b illustrates the real-time risk change for UAVs 0–18 during a single inference run in a large-scale scenario. Within the first 20 steps, the reconnaissance UAVs were the first to be detected, causing their risk to rise rapidly; between steps 20 and 60, the strike UAVs penetrated under EW cover, resulting in a dispersed risk distribution. After step 80, the risk for most UAVs drops to a low level, indicating that the surviving UAVs have exited the high-risk zone. The temporal distribution of the risk curve reflects the tactical strategy learned by SAGA.

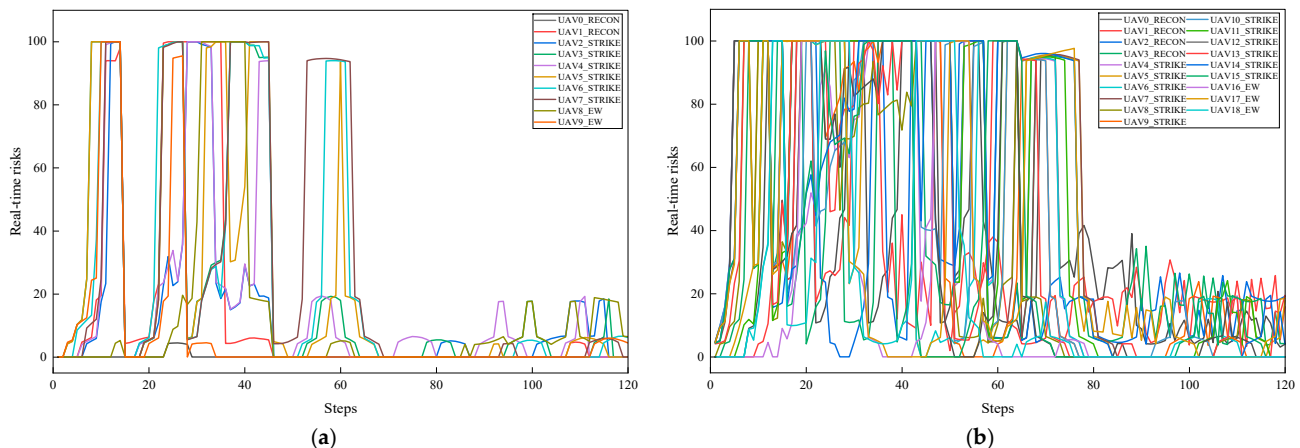


Figure 12. The real-time risks in large-scale and small-scale scenarios. (a) Small-scale. (b) Large-scale.

Figure 13 shows the Gantt charts of the SAM’s status, which visually demonstrates the critical role of EW UAVs in the mission. The frequent occurrence of the purple “JAMMED” status causes the SAM’s detection and tracking capabilities to decline and become intermittent, and this status immediately precedes “DESTROYED.” This demonstrates that SAGA has learned the tactical strategy of “suppression first, destruction later”.

This temporal collaboration between roles is the core tactic of heterogeneous multi-UAVs, validating the practical effectiveness of the algorithm proposed in this paper.

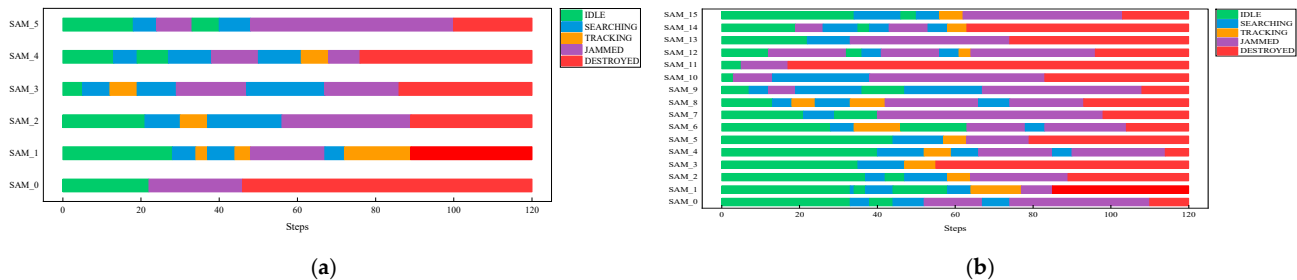


Figure 13. The Gantt charts of the SAM's status in small-scale and large-scale scenarios. (a) Small-scale. (b) Large-scale.

6.3. Generalization Analysis

To verify the generalization capability of the proposed algorithm across different operational scenarios, we conducted tests under two typical defensive deployment modes for SAMs. (1) Fan-shaped deployment: SAMs are deployed in a fan-shaped formation along the primary operational axis for defending the ground target against incoming UAVs, as shown in Figure 14a. (2) Transverse deployment: SAMs are deployed transversely along the primary operational axis for defending the ground target against incoming UAVs, as shown in Figure 14b. Models trained in large-scale scenarios were directly deployed in these two new environments to evaluate their adaptive mission planning capabilities for unknown defense configurations. During the experiments, the number of UAVs, SAMs, and ground targets remained constant. Only the SAM defense deployment modes were adjusted. Each scenario was tested over 100 rounds.

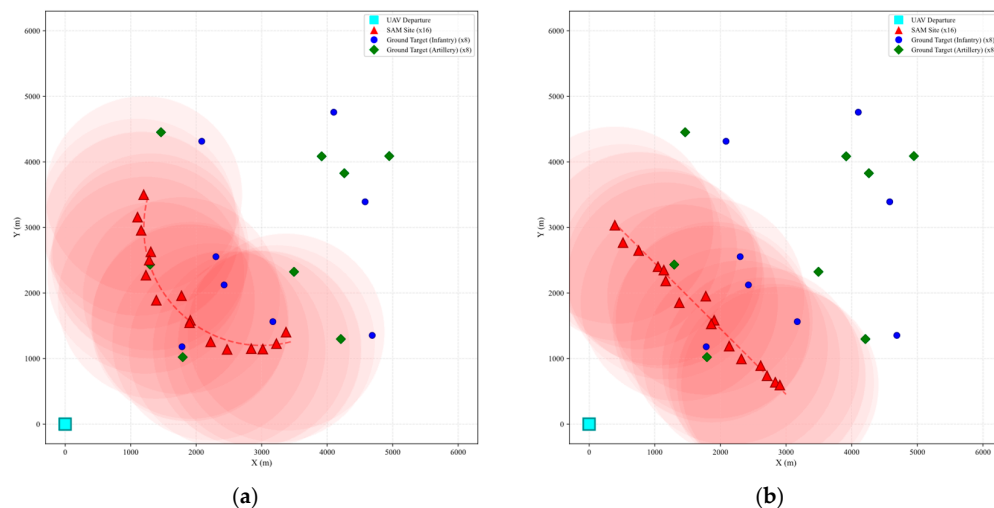


Figure 14. Generalized experimental scenarios. (a) Fan-shaped deployment. (b) Transverse deployment. The red circles indicate the range of the SAMs.

Tables 5 and 6 shows the average values of various metrics for the model under two unknown scenarios. These results validate the generalization ability of the algorithm proposed in this paper. When the trained model was deployed directly to two unknown scenarios for testing, SAGA continued to demonstrate good performance and stability. In contrast, the metrics for each baseline algorithm were lower than those of SAGA and exhibited greater variability, indicating that SAGA has better generalization capabilities.

This also demonstrates that the framework possesses greater robustness and can maintain high operational effectiveness across different scenarios.

Table 5. Generalization experiment metrics (fan-shaped deployment), where the average number of SAMs lost is metric 1, the average number of ground targets lost is metric 2, the average number of UAVs lost is metric 3, and the average reward is metric 4.

Metrics	SAGA	MADDPG	MATD3	QMIX	MAPPO	TarMAC	DGN	G2ANet
Metric 1	15.92 ± 0.63	12.03 ± 2.96	9.03 ± 1.72	6.17 ± 4.02	10.44 ± 2.11	9.52 ± 0.77	7.08 ± 2.21	5.87 ± 2.70
Metric 2	15.94 ± 0.33	11.70 ± 2.45	13.67 ± 1.89	3.07 ± 1.92	9.35 ± 2.24	6.62 ± 3.06	7.98 ± 2.75	4.36 ± 3.27
Metric 3	4.31 ± 0.66	5.78 ± 0.51	9.37 ± 1.57	10.64 ± 0.84	5.78 ± 2.07	7.77 ± 1.94	11.42 ± 1.53	8.58 ± 0.79
Metric 4	1056.97 ± 36.97	436.26 ± 178.95	256.82 ± 115.31	138.83 ± 230.16	847.54 ± 47.92	404.22 ± 101.70	184.53 ± 33.25	127.86 ± 84.26

Table 6. Generalization experiment metrics (transverse deployment), where the average number of SAMs lost is metric 1, the average number of ground targets lost is metric 2, the average number of UAVs lost is metric 3, and the average reward is metric 4.

Metrics	SAGA	MADDPG	MATD3	QMIX	MAPPO	TarMAC	DGN	G2ANet
Metric 1	15.93 ± 0.34	14.70 ± 2.54	9.23 ± 1.78	8.07 ± 3.46	8.84 ± 1.86	9.38 ± 2.31	10.96 ± 3.58	7.75 ± 2.97
Metric 2	15.74 ± 0.48	14.57 ± 1.36	13.06 ± 1.75	3.76 ± 1.73	11.33 ± 0.87	10.64 ± 1.85	9.20 ± 1.08	8.04 ± 2.04
Metric 3	4.04 ± 0.57	6.11 ± 1.20	9.91 ± 1.77	10.86 ± 0.77	9.73 ± 1.82	8.55 ± 0.67	11.07 ± 0.95	11.15 ± 1.66
Metric 4	984.65 ± 119.49	626.81 ± 159.82	256.27 ± 121.62	113.28 ± 194.77	211.74 ± 184.29	276.46 ± 99.37	255.78 ± 188.24	124.57 ± 133.43

6.4. Ablation Experiment

To evaluate the individual contributions of the improved modules in our proposed SAGA framework, we conducted a series of ablation experiments. These experiments primarily validated the effectiveness of the GNN network, the edge predictor, and the sparsity penalty. We compared the following algorithm variants:

- (1) SAGA: The complete algorithmic framework proposed in this paper.
- (2) w/o GNN: SAGA with the GNN module removed (SAGA-MLP), to verify whether graph message passing contributes to collaborative decision-making.
- (3) w/o Edge Predictor: SAGA with the edge predictor removed (SAGA-FC), using a fully connected graph with no learnable sparse communication topology. This verifies the value of the edge predictor.
- (4) w/o Sparsity Loss: SAGA with the sparsity penalty removed (SAGA-NS), to verify whether the L1 sparsity constraint promotes efficient communication.

All experiments were conducted in a small-scale scenario.

Figure 15 compares the training convergence curves of SAGA and its ablation variants in small-scale scenario. The figure clearly illustrates the differences in the contributions of each module to training convergence performance. The shaded area represents the standard deviation.

SAGA demonstrates the best convergence performance among all variants. This is attributed to the sparse communication topology, filtered by the edge predictor via the GATv2 graph attention mechanism, which enables multi-round distributed consensus. This allows heterogeneous agents to efficiently exchange critical tactical information in partially observable environments, thereby accelerating the learning of collaborative strategies.

SAGA-NS's convergence performance is second only to that of the full SAGA model, with the final reward stabilizing in the range of approximately 340–360, a decrease of about 8–13% compared to the full SAGA model. This indicates that even without the sparsity penalty, the edge predictor itself is still capable of learning an effective communication topology. However, the absence of sparsity penalties means that communication links lack the explicit ability to eliminate redundancy, some redundant connections are retained in the topology, and the presence of low-value communication links dilutes the weight of

critical information, resulting in slightly lower strategy learning efficiency compared to full SAGA.

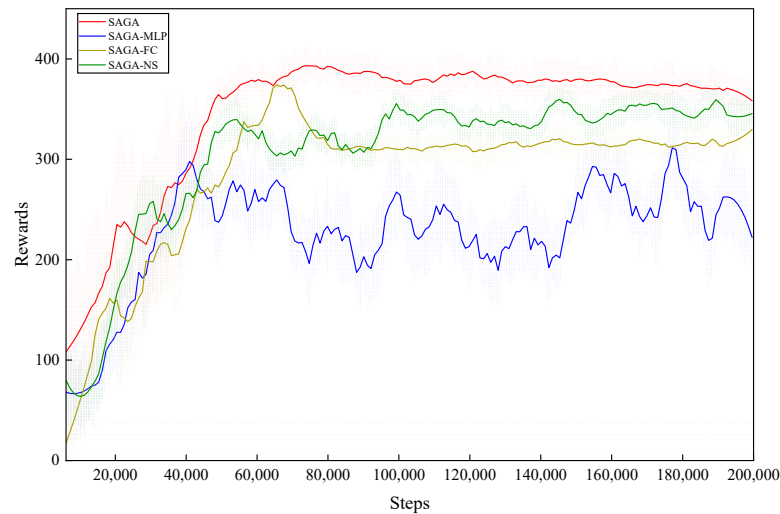


Figure 15. Comparison of algorithm training reward curves in ablation experiment.

The convergence curve of SAGA-FC eventually stabilizes at approximately 310, due to the impact of the fully connected communication topology on the GATv2 attention mechanism. The model struggles to effectively focus attention on a small number of high-value communication links via gradient descent; it cannot distinguish between important and redundant information, which is equivalent to distributing the information aggregation weights evenly across all communication links. This causes GNN message passing to degrade into simple mean pooling, losing the selective advantage of the graph attention mechanism.

SAGA-MLP performed the worst and exhibited significant volatility, with its convergence curve lacking a clear upward trend. The MLP encoder independently encodes each UAV's observations into policy features and lacks the ability for explicit communication between agents. In certain partially observable air-to-ground missions, this structure forces each UAV to make independent decisions based solely on its own local observations, without access to information from allied units, causing the overall strategy to degenerate into independent behavior. This result directly validates the importance of the GNN graph reasoning module.

To further elucidate the differences in information aggregation behavior across different communication topologies, we introduce attention weight entropy as a quantitative metric to measure the degree of attention focus in communication. The attention entropy of a target UAV i_2 is defined as the information entropy of all attention weights received by that node from source nodes:

$$H_{i_2} = - \sum_{i_1 \in N(i_2)} \alpha_{i_1 i_2} \cdot \log(\alpha_{i_1 i_2}) \quad (73)$$

where $\alpha_{i_1 i_2}$ represents the normalized attention weight from the source UAV i_1 to the target UAV i_2 , and $N(i_2)$ represents the set of source nodes sending messages to i_2 . This metric reflects the degree to which the attention distribution of each UAV is concentrated when aggregating information.

Figure 16 shows heatmaps comparing the attention weights of the SAGA, SAGA-FC, and SAGA-NS in small-scale scenarios. The rows and columns of each heatmap represent the attention weights of the source UAV i_1 toward the target UAV i_2 . Darker colors

indicate higher weights. Figure 16a reveals a distinct sparse clustering pattern: a small number of key UAVs, such as the two reconnaissance UAVs (UAV2→UAV1) and the two jamming UAVs, receive high attention weights. The transmission of target instructions from reconnaissance UAVs to strike UAVs, as well as coordination among strike UAVs, dominates the aggregated information at the target nodes. Attention weights for non-critical links are lower, appearing in lighter colors. This phenomenon indicates that the synergistic effect of the edge predictor and sparsity penalty effectively prunes redundant communication links. The edge predictor filters out communication edges with mission value, while the sparsity penalty further reduces the weights of low-value links, enabling each UAV to concentrate its limited attention resources on the few friendly nodes most relevant to its own sub-task, thereby achieving efficient selective information aggregation.

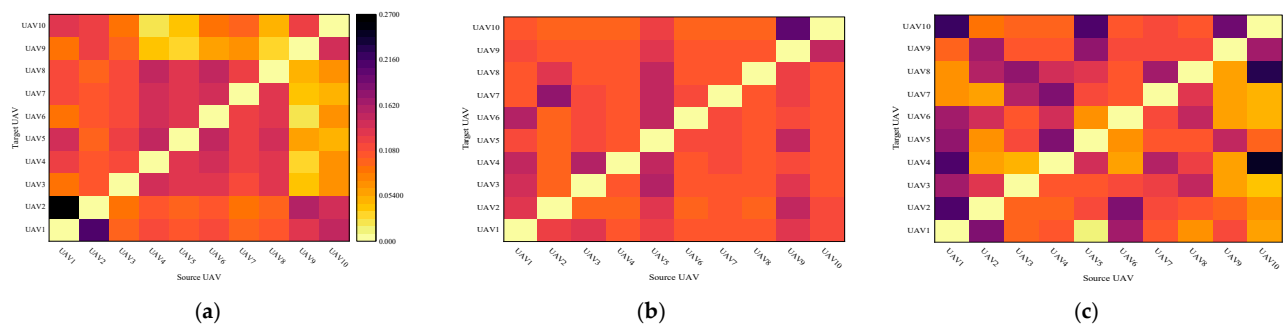


Figure 16. Attention weight comparison heatmap. (a) SAGA. (b) SAGA-FC. (c) SAGA-NS.

Since SAGA-FC lacks an edge predictor for link filtering, communication edges exist between all UAV pairs, and the entire heatmap appears as a uniform dark region. This indicates that each UAV treats messages from all friendly nodes nearly equally during information aggregation, making it prone to overlooking critical information. SAGA-NS retains the link filtering capability of the edge predictor, resulting in an attention distribution with a certain degree of selectivity. Certain key UAV pairs receive higher attention weights, forming distinguishable hotspots. However, due to the absence of L1 sparsity constraints, a certain amount of redundant information is still mixed into the aggregated data. It is worth noting that the non-uniform distribution of attention weights in SAGA-NS also confirms that the edge predictor itself possesses a certain ability to discern link value, the role of the sparsity penalty is to further optimize the model's information filtering capability on this basis.

Table 7 presents the statistical results of effectiveness metrics for the four algorithms across 100 test rounds. SAGA achieved the optimal values across all core metrics, further validating the necessity of each module. In summary, the GNN graph reasoning module provides UAVs with collaborative communication capabilities. By learning sparse communication topologies, the edge predictor effectively avoids the attention dilution problem associated with fully connected graphs, ensuring stable convergence during training and the effectiveness of the strategy. As an auxiliary constraint, the sparsity penalty applies additional pruning pressure based on the link selection performed by the edge predictor, enabling the model to achieve better collaborative results with fewer communication resources and ensuring communication efficiency. The collaborated interaction of these three components constitutes the complete SAGA framework.

Table 7. Ablation experiment metrics, where the average number of SAMs lost is metric 1, the average number of ground targets lost is metric 2, and the average number of UAVs lost is metric 3.

Metrics	SAGA	SAGA-MLP	SAGA-FC	SAGA-NS
Metric 1	5.90 ± 0.31	3.74 ± 0.65	2.93 ± 1.37	3.02 ± 0.74
Metric 2	7.86 ± 0.35	4.44 ± 0.47	4.02 ± 0.82	5.11 ± 1.19
Metric 3	1.18 ± 0.41	4.12 ± 0.73	3.67 ± 0.55	4.52 ± 0.50

To quantitatively assess communication efficiency brought by SAGA’s learned sparse topology, we adopt three evaluation metrics:

1. Average messages per step: average count of physical transmitted messages over one simulation timestep.
2. Retained-edge ratio: ratio of active learned logical edges compared with fully connected complete graph. Lower value indicates sparser topology.

We also record average reward, and test SAGA, SAGA-FC, and SAGA-NS. The test results are shown in Table 8.

Table 8. Comparison of quantitative evaluation results for communication efficiency.

Metrics	SAGA	SAGA-FC	SAGA-NS
Average messages per step	21.3	71.2	44.7
Retained-edge ratio (%)	19.6	100	42.1
Average reward	394.6	311.3	348.2

SAGA-FC uses a full fully connected graph, and every agent attempts to communicate with all teammates, yielding 71.2 messages per step and a 100% edge ratio, but its reward is low. SAGA-NS uses an edge predictor but removes the L1 sparsity penalty; it partially prunes redundant edges, yet still keeps many low-value links. Message count is higher than that of SAGA. SAGA with the edge predictor plus the L1 sparsity penalty yields the lowest average message count, only retaining 19.6% of all possible directed edges. Meanwhile it achieves highest average reward. This demonstrates SAGA can drastically cut communication overhead without sacrificing mission performance.

7. Conclusion and Future Work

This paper proposes SAGA, a learnable sparse attention graph architecture suitable for heterogeneous multi-UAV air-to-ground mission planning. This framework provides solutions to the challenges of partial observability, heterogeneous UAV collaboration, and efficient communication in air-to-ground missions. This architecture deeply integrates graph reasoning with policy optimization. The GATv2-based graph neural network encoder achieves multi-round distributed consensus among UAVs through a learnable attention mechanism, enabling each agent to selectively aggregate tactical information from its communication neighbors. An MLP edge predictor, trained jointly with the policy, generates a sparse communication topology by predicting the value of each potential communication link, thereby avoiding the attention dilution problem found in fully connected graphs. Additionally, an L1 communication sparsity penalty term is introduced into the training objectives, enabling the model to achieve optimal collaboration with fewer communication resources. Across all evaluation metrics, the SAGA model consistently outperforms the baseline algorithms. Generalization experiments further validate the framework’s robustness. When models trained on standard large-scale scenarios are directly deployed to unseen environments, SAGA continues to lead across all metrics. The results demonstrate that the sparse communication topologies and collaborative strategies learned by the model can adapt to mission configurations beyond the training set.

Ablation experiments further validate the effectiveness of each component. In summary, the SAGA framework provides an intelligent, efficient, and theoretically sound solution for heterogeneous multi-UAV air-to-ground mission planning in partially observable combat environments.

A limitation of this study is the discrepancy between simulation assumptions and practical deployment. Future work will address this by making the following improvements to support real-world deployment: (1) designing a lightweight encoder adapted for real-time inference to ensure acceptable inference latency; (2) incorporating diverse noise and parameter perturbations into the simulation to enhance the model's robustness; and subsequently, integrating real-world data into algorithm training to bridge the gap between simulation and real-world deployment; (3) introducing a physical cost term into the loss function to replace the simple L1 sparsity penalty, thereby better aligning with real-world scenarios; and (4) exploring dynamic topology-adaptive learning methods in communication-denied environments, incorporating more refined adversary modeling, and extending this framework to larger-scale hybrid formation collaboration scenarios. Furthermore, future work will further investigate the integration of an event-triggered intermittent sampling coordination mechanism into SAGA to address the issues of asynchronous perception and temporary communication interruptions in real-world multi-UAV deployment scenarios. At the same time, real-world physical-layer communication constraints, such as interference, transmission energy consumption, and bandwidth limitations, will be introduced. By combining SAGA's task-aware sparse logical topology learning with existing joint frameworks for trajectory, power, and resource allocation in UAV network research, we aim to bridge the gap between multi-agent decision-making at the task level and the characteristics of real-world wireless communications.

Author Contributions: Conceptualization, H.S. and X.G.; methodology, H.S., X.G. and G.W.; software, H.S.; validation, H.S., X.G. and X.B.; formal analysis, H.S. and X.B.; investigation, H.S.; resources, G.W.; data curation, H.S.; writing—original draft preparation, H.S. and X.B.; writing—review and editing, H.S., X.B. and X.G.; visualization, H.S.; supervision, G.W.; project administration, X.G.; funding acquisition, X.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data available on request due to privacy restrictions. The data presented in this study are available on request from the corresponding author due to necessary approvals for privacy protection.

DURC Statement: This research is limited to simulation studies of air-to-ground mission planning for multi-UAV systems. It is intended to advance the theory and technology of collaborative mission planning for multi-UAV systems and does not pose a threat to public health or national security. The authors acknowledge that research involving UAV mission planning has dual-use potential and confirm that all necessary safeguards have been implemented to mitigate the risk of misuse of the technology. Specifically, this study presents only an algorithmic framework for simulation verification, it does not provide any practical guidance regarding weapon systems or operational applications, and all necessary precautions have been taken to prevent potential misuse. As an ethical responsibility, authors strictly adhere to relevant national and international laws about DURC. Authors advocate for responsible deployment, ethical considerations, regulatory compliance, and transparent reporting to mitigate misuse risks and foster beneficial outcomes.

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

Appendix A. Notation Table

Symbol	Description
N	set of heterogeneous UAV agents
S	global state space of Dec-POMDP
O_i	local observation space
A_i	hybrid action space
R	team-shared reward
Z	observation function
h_i	altitude
ψ_i	heading angle
v_i	speed
a_i^{cont}	continuous action vector
a_i^{disc}	discrete action vector
a_i^{tgt}	target engagement selection
a_i^{def}	defense mode selection
a_i^{comm}	communication broadcast flag
f_i	fuel level
am_i	ammunition count
p_k^{SAM}	position of SAM site
R_{max}	maximum detection range of UAV
$P_d(R, \sigma)$	probability of detection
d_{ik}	detection range of UAV RWR
threat_{ik}	threat level
risk_i	aggregate risk score
R_{eff}	RCS-scaled effective range of SAM
$R_{\text{engage}}^{\text{SAM}}$	engagement range of SAM
Q_{track}	tracking quality of SAM

Appendix B. Analysis of Algorithm Time Complexity

Appendix B.1. Symbol Description

Symbol	Description
N	Number of agents
d_o	Observation vector dimension
d_h	Encoder hidden layer dimensions
d_n	Node feature dimensions
d_e	Edge feature dimensions
d_p	Dimensions of the hidden layer in the edge predictor
K	Number of attention layers
A	Number of attention heads
E	Number of communication edges
d_c	Continuous action dimension
d_d	Sum of the discrete action dimensions
T	Number of time steps within a single minibatch
E_p	Number of PPO update cycles
M	Number of minibatches

Appendix B.2. Forward Pass Complexity

The GNN encoder described in this paper consists of three components: an observation encoder, an edge predictor, and K layers of graph attention convolutions. The observation encoder is a three-layer MLP. For a single sample, the computational cost is:

$$T_{\text{enc}} = O(d_o \cdot d_h + d_h^2 + d_h \cdot d_n)$$

The edge predictor performs pairwise predictions for all unordered node pairs (i, j) . Its input is a concatenation of the features of the two nodes, and it outputs the probability that an edge exists after passing through a three-layer MLP. The cost for each node pair is $O(d_n \cdot d_p + d_p^2)$, and the total number of node pairs is $N(N-1)/2$, so:

$$T_{\text{ep}} = O(N^2 \cdot (d_n \cdot d_p + d_p^2))$$

Each GATv2 layer in the graph attention convolutional layer includes linear projections of the source, target, and edges, as well as edge attention computation, message aggregation, and output projection. The source, target, and output projections each have a complexity of $O(N \cdot d_n^2)$. The edge projection has a complexity of $O(E \cdot d_e \cdot d_n)$. Attention and message aggregation are performed along E edges and A heads, at a cost of $O(E \cdot d_n)$. Therefore, the cost of a single layer is:

$$T_{\text{gat}} = O(N \cdot d_n^2 + E \cdot d_n)$$

Therefore, the cost of a single forward pass is:

$$T_{\text{gnn}} = O(N \cdot (d_o \cdot d_h + d_h^2 + d_h \cdot d_n) + N^2 \cdot (d_n \cdot d_p + d_p^2))$$

Appendix B.3. Inference and Training Complexity

Single step reasoning organizes the observations of N agents into a graph for forward propagation and obtains actions by sampling from a continuous distribution with probability $O(N \cdot d_n \cdot d_c)$ and a discrete distribution with probability $O(N \cdot d_n \cdot d_d)$. Therefore, the complexity of single step reasoning is

$$T_{\text{inf}} = O(N \cdot (d_o \cdot d_h + d_h^2 + d_h \cdot d_n + d_n \cdot d_c + d_n \cdot d_d) + N^2 \cdot (d_n \cdot d_p + d_p^2) + K \cdot (N \cdot d_n^2 + E \cdot d_n))$$

Since training requires reconstructing the communication graph and computing edge probabilities at each time step, the encoder is executed T times within each minibatch.

A single PPO update consists of E_p rounds, with M minibatches per round. Therefore, the complexity of a single update is:

$$T_{\text{train}} = O(E_p \cdot M \cdot T \cdot (N \cdot (d_o \cdot d_h + d_h^2 + d_h \cdot d_n) + N^2 \cdot (d_n \cdot d_p + d_p^2) + K \cdot (N \cdot d_n^2 + E \cdot d_n)))$$

References

- Gu, X.; Zhang, G. A survey on UAV-assisted wireless communications: Recent advances and future trends. *Comput. Commun.* **2023**, *208*, 44–78.
- Wang, J.; Jia, G.; Yu, K.; Zhang, Y.; Guo, Z.; Hou, Z. A hybrid evolutionary algorithm integrated with specific neighborhood search for collaborative task planning of heterogeneous multi-UAVs under SEAD scenarios. *Proc. Inst. Mech. Eng. Part G J. Aeronaut. Eng.* **2025**, *239*, 403–439.
- Yue, L.; Yang, R.; Zuo, J.; Zhang, Y.; Li, Q.; Zhang, Y. Unmanned Aerial Vehicle Swarm Cooperative Decision-Making for SEAD Mission: A Hierarchical Multi-Agent Reinforcement Learning Approach. *IEEE Access* **2022**, *10*, 92177–92191.
- Zhang, J.; Xing, J. Cooperative task assignment of multi-UAV system. *Chin. J. Aeronaut.* **2020**, *33*, 2825–2827.
- Ding, Z.; Huang, T.; Lu, Z. Learning Individually Inferred Communication for Multi-Agent Cooperation. In *Advances in Neural Information Processing Systems*; NeurIPS: Paris, France, 2020; Volume 33, pp. 22069–22079.

6. Zhao, Y.; Qi, H.; Gao, X.; Zhu, R. A two-stage approach for multi-UAV mission planning in vessel logistics services. *Chin. J. Aeronaut.* **2026**, *in press*.
7. Aramyan, M.; Manucharyan, A.; Poghosyan, L.; Bakaryan, T.; Hovakimyan, N. Game-Theoretic Coordination for Time-Critical Missions of UAV Systems. *IEEE Trans. Control Netw. Syst.* **2026**, *early access*.
8. Yang, Y.; Polycarpou, M.M.; Minai, A.A. Multi-UAV Cooperative Search Using an Opportunistic Learning Method. *J. Dyn. Syst. Meas. Control* **2007**, *129*, 716–728.
9. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; Petersen, S. Human-Level Control Through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533.
10. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning. *Nature* **2019**, *575*, 350–354.
11. Zhang, X.; Yang, B.; Cao, X.; Yu, Z.; Alexandropoulos, G.C.; Zhang, Y.; Debbah, M.; Yuen, C. Multi-Agent Deep Reinforcement Learning for Safe Autonomous Driving with RICS-Assisted MEC. *IEEE Trans. Intell. Transp. Syst.* **2026**, *27*, 800–815.
12. Oliehoek, F.A.; Spaan, M.T.J.; Vlassis, N. Optimal and Approximate Q-Value Functions for Decentralized POMDPs. *J. Artif. Intell. Res.* **2008**, *32*, 289–353.
13. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*; NeurIPS: Paris, France, 2017; Volume 30, pp. 6379–6390.
14. 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 International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 10–15 July 2018; pp. 4295–4304.
15. Yu, C.; Velu, A.; Vinitzky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In *Advances in Neural Information Processing Systems*; NeurIPS: Paris, France, 2022; Volume 35, pp. 24611–24624.
16. Das, A.; Gervet, T.; Romoff, J.; Batra, D.; Parikh, D.; Rabbat, M.; Pineau, J. TarMAC: Targeted Multi-Agent Communication. In *Proceedings of the International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, 9–15 June 2019; pp. 1538–1546.
17. Battaglia, P.W.; Hamrick, J.B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gulcehre, C. Relational Inductive Biases, Deep Learning, and Graph Networks. *arXiv* **2018**, arXiv:1806.01261.
18. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France, 24–26 April 2017.
19. Zhang, H.; Gan, X.; Li, S.; Chen, Z. UAV safe route planning based on PSO-BAS algorithm. *J. Syst. Eng. Electron.* **2022**, *33*, 1151–1160.
20. Jiao, Y.; Fu, W.; Cao, X.; Kou, K.; Tang, J.; Shen, R.; Zhang, Y.; Du, H. A Cooperative Decision-Making and Control Algorithm for UAV Formation Based on Non-Cooperative Game Theory. *Drones* **2024**, *8*, 698. <https://doi.org/10.3390/drones8120698>.
21. Yang, J.; You, X.; Wu, G.; Hassan, M.M.; Almogren, A.; Guna, J. Application of Reinforcement Learning in UAV Cluster Task Scheduling. *Future Gener. Comput. Syst.* **2019**, *95*, 140–148.
22. Yang, X.X.; Yao, W.Q.; Zhang, Y.; Yu, H.; Wang, C. Multi-UAV Cooperative Search in Partially Observable Low-Altitude Environments Based on Deep Reinforcement Learning. *Drones* **2025**, *9*, 825.
23. Chen, R.; Hu, F. CaST-MASAC: Integrating Causal Inference and Spatio-Temporal Attention for Multi-UAV Cooperative Task Planning. *Drones* **2025**, *9*, 644.
24. Liang, Q.; Yang, Y.; Liang, S.; Li, H. SA-DSM-MADDPG: Multi-UAV Cooperative Encirclement Based on Dual Stochastic Mechanisms. *Drones* **2026**, *10*, 360.
25. Zhang, Y.; Liu, L.; Wang, G. UAV Cooperative Air Combat Maneuver Decision Based on Multi-Agent Reinforcement Learning. *J. Syst. Eng. Electron.* **2022**, *33*, 1042–1052.
26. Guo, Q.; Liu, H.; Tian, Y.; Sun, Y.; Yu, Y. A hierarchical hybrid action reinforcement learning framework for beyond-visual-range air combat decision-making. *Def. Technol.* **2026**, *in press*.
27. Li, B.; Chen, Z.; Wang, F. Joint Optimization of Configuration and Mission Planning for Heterogeneous UAVs in SEAD Scenarios. *J. Natl. Univ. Def. Technol.* **2024**, *46*, 127–136.
28. Jiang, Q.; Yan, Y.; Dai, Y.; Yang, Z.; Cao, H.; Wang, B.; Ma, X. Autonomous Task Planning of Intelligent Unmanned Aerial Vehicle Swarm Based on Deep Deterministic Policy Gradient. *Drones* **2025**, *9*, 272. <https://doi.org/10.3390/drones9040272>.
29. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; Graepel, T. Value-Decomposition Networks for Cooperative Multi-Agent Learning. In *Proceedings of the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, Stockholm, Sweden, 10–15 July 2018; pp. 2085–2087.

30. Jiang, J.; Dun, C.; Huang, T.; Lu, Z. Graph Convolutional Reinforcement Learning. In Proceedings of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, 26 April–1 May 2020.
31. Liu, Y.; Wang, W.; Hu, Y.; Hao, J.; Chen, X.; Gao, Y. Multi-Agent Game Abstraction via Graph Attention Neural Network. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 7211–7218.
32. Niu, Y.; Paleja, R.; Gombolay, M. Multi-Agent Graph-Attention Communication and Teaming. In Proceedings of the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS), London, UK, 3–7 May 2021; pp. 964–972.
33. Li, S.; Gupta, J.K.; Morales, P.; Allen, R.; Kochenderfer, M.J. Deep Implicit Coordination Graphs for Multi-Agent Reinforcement Learning. In Proceedings of the International Conference on Autonomous Agents and Multi Agent Systems (AAMAS), London, UK, 3–7 May 2021; pp. 764–772.
34. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
35. Brody, S.; Alon, U.; Yahav, E. How Attentive Are Graph Attention Networks? In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022.
36. Sukhbaatar, S.; Szlam, A.; Fergus, R. Learning Multiagent Communication with Backpropagation. In *Advances in Neural Information Processing Systems*; NeurIPS: Paris, France, 2016; Volume 29, pp. 2244–2252.
37. Singh, A.; Jain, T.; Sukhbaatar, S. Learning When to Communicate at Scale in Multiagent Cooperative and Competitive Tasks. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
38. Kim, D.; Moon, S.; Hostallero, D.; Kang, W.J.; Lee, T.; Son, K.; Yi, Y. Learning to Schedule Communication in Multi-Agent Reinforcement Learning. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
39. That, V.; Muy, S.; Lee, J.-R. Multi-UAV-aided power-up and data collection. *Expert Syst. Appl.* **2025**, *297*, 129401.
40. Muy, S.; Ron, D.; Lee, J.-R. Reducing E2E delay in wireless multi-hop networks: A multi-agent DRL-based approach. *KSII Trans. Internet Inf. Syst.* **2025**, *19*, 1607–1624.
41. Muy, S.; Lee, J.-R. Joint optimization of trajectory, beamforming, and power allocation in UAV-enabled WPT networks using DRL combined with water-filling algorithm. *Veh. Commun.* **2023**, *43*, 100632.
42. Ge, X.; Han, Q.; Zhang, X.; Ding, D.; Ning, B. Distributed coordination control of multi-agent systems under intermittent sampling and communication: A comprehensive survey. *Sci. China Inf. Sci.* **2025**, *68*, 151201.

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.