ACORN: Acyclic Coordination with Reachability Network to Reduce Communication Redundancy in Multi-Agent Systems

Yi Xie
FAET, Fudan University
Shanghai, China
yixie22@m.fudan.edu.cn

Ziqing Zhou
FAET, Fudan University
Shanghai, China
21110860021@m.fudan.edu.cn

Chun Ouyang*
FAET, Fudan University
Shanghai, China
oy_c@fudan.edu.cn

Siao Liu
FAET, Fudan University
Shanghai, China
saliu20@fudan.edu.cn

Linqiang Hu
FAET, Fudan University
Shanghai, China
18110860018@fudan.edu.cn

Zhongxue Gan*
FAET, Fudan University
Shanghai, China
ganzhongxue@fudan.edu.cn

ABSTRACT

Effective communication is essential in multi-agent reinforcement learning (MARL) for coordinating actions and maximizing collective rewards. Two common approaches for establishing communication are Graph Neural Networks (GNNs) and Transformers. Both methods introduce communication redundancy in complex scenarios. GNN-based methods model agent relationships through entire graph structures, leading to increased computational time. Transformers also increase computations due to self-attention calculations at each node. In this study, the *ACORN* (*Acyclic Coordination with Reachability Networks*) framework was introduced, utilizing acyclic coordination combined with a reachability-based attention mechanism. The most relevant nodes and connections in the GNN graph are used for self-attention calculations. Time complexity is reduced to $O(|V| \times nk \times d)$, which is significantly better than the $O(|V|^2 d)$ complexity of standard Transformers. Acyclicity is ensured through Auto-Regressive Policy Learning and Sequence-Based Critic Learning. Experiments demonstrate that ACORN outperforms state-of-the-art methods, achieving an average improvement of 11% over MAT in challenging SMACV2 tasks and a 17% improvement within the same training time and steps.

KEYWORDS

Reinforcement Learning; Multi-agent Systems; Communication Topology; Graph Neural Network

ACM Reference Format:
Yi Xie, Ziqing Zhou, Chun Ouyang*, Siao Liu, Linqiang Hu, and Zhongxue Gan*. 2025. ACORN: Acyclic Coordination with Reachability Network to Reduce Communication Redundancy in Multi-Agent Systems. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

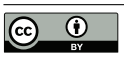
1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has emerged as a significant research area, allowing multiple agents to learn and make decisions in complex, dynamic environments[21, 31, 42]. Communication between agents is recognized as critical for enhancing overall performance[6]. Effective cooperation among agents are considered crucial in the fields of autonomous driving [10], traffic signal control [27], and collaborative robotics [3]. However, as the number of agents increases, the complexity of their interactions is heightened, leading to potential communication inefficiencies [20]. One major challenge is identified as communication redundancy, which occurs when agents exchange overlapping or unnecessary information, resulting in repeated computations [19, 37].

Existing MARL methods primarily utilize Graph Neural Networks (GNNs) [5, 14, 15, 33, 38] and Transformers [9, 36] to facilitate communication among agents. While these methods improve agent communication, they also introduce redundancy in complex scenarios[13]. GNN-based methods model agent relationships through entire graph structures, resulting in increased computational time and communication redundancy due to edge updates and aggregation [22, 26]. Transformer-based architectures employ self-attention mechanisms that necessitate computations for each agent's relationships, leading to higher computational time and resource consumption [32, 36]. These limitations highlight the need for a more efficient coordination framework. The detailed challenges of these methods are discussed in Section 3.

To address communication redundancy, *ACORN* (Acyclic Coordination with Reachability Networks) was proposed. First, a communication graph is constructed based on acyclic reachability, utilizing agent relationships to form Temporal Acyclic Graphs (TAGs). Then, a theoretically validated graph-to-matrix conversion method is developed, projecting the communication graph into a matrix representation within the admissible solution set of acyclic graphs. Finally, a reachability-based attention mechanism is implemented by leveraging the communication matrix. It was utilized for developing an auto-regressive policy learning framework with a sequence-based critic strategy to facilitate effective inter-agent coordination. This framework minimizes communication redundancy by concentrating computations on crucial nodes and connections.

Experiments were conducted to validate performance and compare wall time usage against state-of-the-art (SOTA) methods on various benchmark datasets, including SMAC, SMACV2, GRF, and MAMUJOCO. Additionally, ablation studies on each component demonstrated that the algorithm's improvement is attributed to the



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

TAG-based reachability attention mechanism. The main contributions are summarized as follows:

- Communication redundancy in GNN-based MARL methods are theoretically explained, and the trade-off between accuracy and computational efficiency between Transformers and GNNs is empirically demonstrated.
- ACORN, an acyclic coordination framework, is introduced to reduce computational complexity through a reachability-based attention mechanism and sequential learning.
- Extensive experiments across five challenging benchmarks demonstrate that ACORN significantly outperforms SOTA MARL methods, validating its effectiveness.

2 RELATED WORK

Cooperative MARL methods, including value decomposition and policy gradient approaches, are established as essential baselines for multi-agent coordination. **QMIX** [26] decomposes the joint action-value into individual Q-values through a mixing network, facilitating efficient cooperative learning without explicit communication. **MAPPO** [39] extends Proximal Policy Optimization to multi-agent settings by sharing policy parameters and employing centralized training with decentralized execution. **HAPPO** [11] introduces hierarchical PPO, which divides policies into higher and lower-level components to enhance coordination and learn complex behaviors. These methods operate without explicit communication among agents, encountering challenges related to non-stationarity and scalability in complex environments [9]. Deep coordination graphs utilize static connections between all agent pairs [17] in GNNs, providing high representational capacity for centralized Q-values while introducing computational challenges during execution. To address this, state-dependent graphs are proposed, allowing each state to have a unique coordination structure. For example, **DICG** [14] employs attention mechanisms to learn message-dependent coordination graph structures with soft edge weights. **CASEC** [33] constructs context-aware sparse topologies based on the variance of payoff functions, while **SOPCG** [38] leverages dynamic graph topologies and structured graph classes to enhance accuracy and efficiency. As for Transformers, attention mechanisms are also applied in MARL to enhance communication. **MAAC** [35] models a centralized critic using attention, from which decentralized actors are derived through soft actor–critic. **ATOC** [23] determines whether agents should communicate with neighbors using an attentional communication model. While these methods enable agents to treat neighbors differently, limitations arise due to non-stationarity [25]. **A2PO** [34] adopts a sequential scheme that updates policies agent-by-agent, demonstrating strong performance. Auto-regressive policy learning [8] generates each agent's action based on its observation and the actions of preceding agents in a specified order. Similarly, **MAT** [36] employs an encoder-decoder architecture to transform multi-agent joint policy optimization into a sequence modeling process. Despite their effectiveness, these methods often face unavoidable redundancy.

3 MOTIVATION

An analysis of various GNNs used to model multi-agent communication was conducted, as shown in Table 1 [5, 15, 38]. The graph structures reveal that communication redundancy manifests in two primary forms: Edge Update Redundancy (ER) and Aggregation Redundancy (AR). ER is observed when multiple target agents share the same source agent, leading to repeated computations during edge updates. AR is identified when agents with common neighbors perform redundant aggregation calculations. GNN-based methods are particularly susceptible to significant communication redundancy in complex modeling scenarios, which negatively impacts performance. Notably, acyclic graphs offer advantages in reducing AR compared to other types of graphs.

Conversely, redundancy issues are alleviated by Transformer-based methods such as MAT [36] and TransfQMix [9] through self-attention mechanisms. However, agent relationships are inherently treated as fully connected, resulting in increased computations. To ensure a fair comparison of the node classification capabilities of GNNs and Transformer methods, which is critical for MARL tasks, all mentioned methods are empirically evaluated on the larger-scale **ogbg-code2** dataset, designed to test node classification capabilities. The experimental results, presented in *Appendix A.1*, indicate that Transformer-based algorithms like SAT [2] achieve higher F1 scores than GAT but require significantly more time. This demonstrates a trade-off between accuracy and computational efficiency in communication between Transformers and GNNs. These challenges inspire further exploration of MARL by combining Transformer architectures with acyclic graph structures.

4 PRELIMINARIES

4.1 Problem Definition

Cooperative MARL are typically modeled as Markov games, defined by the tuple $< N, O, A, R, P, \gamma >$ [16]. Here, $N = \{1, \ldots, n\}$ denotes the set of agents, $O = \prod_{i=1}^{n} O^i$ the joint observation space, and $A = \prod_{i=1}^{n} A^i$ the joint action space. The reward function $R : O \times A \to [-R_{\max}, R_{\max}]$ assigns rewards based on observations and actions, while the transition function $P : O \times A \times O \to \mathbb{R}$ governs state transitions. The discount factor $\gamma \in [0, 1]$ determines the importance of future rewards. At each timestep t, each agent $i \in N$ receives an observation $o_t^i \in O^i$ and selects an action according to its policy π^i, part of the joint policy π. Agents act concurrently without sequential dependencies. The joint policy π and transition function P define the marginal observation distribution $\rho_\pi(o)$. After executing the joint action a_t, the team receives a shared reward $R(o_t, a_t)$ and transitions to the next observation o_{t+1} based on $P(\cdot|o_t, a_t)$. The objective is to maximize the discounted cumulative return $R^\gamma = \sum_{t=0}^{\infty} \gamma^t R(o_t, a_t)$.

4.2 Transformer on Graphs

Transformer models [32] have gained prominence in graph learning due to their ability to capture complex relationships. The standard Transformer architecture comprises a self-attention mechanism followed by a feed-forward network, each integrated with residual connections and normalization layers. Given input node features $\mathbf{X} \in \mathbb{R}^{N \times d}$, the self-attention mechanism computes query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) representations using projection matrices

Table 1: GNN algorithms on neighborhood message passing models and redundancy analysis. σ denotes the activation function, W denotes the model weights, $\mathcal{N}(v)$ denotes the neighbor set of vertex v, $e_{u,v}$ means the weight of edge (u,v), a_v^k represents the aggregation results for vertex v, h_v^k denotes the feature vector of vertex v at the k-th layer, ER, and AR.

Algorithms	EdgeUpdate	Aggregate	VertexUpdate	ER	AR
GCN	Null	$a_v^k = \sum_{u\in\mathcal{N}(v)} h_u^{k-1}$	$h_v^k = \sigma(W^k \cdot a_v^k)$	No	Yes
GIN	Null	$a_v^k = \sum_{u\in\mathcal{N}(v)} h_u^{k-1}$	$h_v^k = \sigma(W^k \cdot a_v^k)$	No	Yes
SGC	Null	$a_v^k = \sum_{u\in\mathcal{N}(v)} h_u^0$	$h_v^k = W^k \cdot a_v^k$	No	Yes
PNA	Null	$a_v^k = \text{Aggregate}\left(\{\text{AGG}(h_u^{k-1}) \mid u \in \mathcal{N}(v)\}\right)$	$h_v^k = \sigma(W^k \cdot [h_v^{k-1} \| a_v^k])$	No	Yes
DAGNN	Null	$a_v^k = \sum_{u\in\mathcal{N}(v)} \alpha_{vu} \cdot h_u^{k-1}$	$h_v^k = \sigma(W^k \cdot [h_v^{k-1} \| a_v^k])$	No	Yes
DAG-GNN	$e_{u,v} = \sigma\left(W_e \cdot [h_u^{k-1} \| h_v^{k-1}]\right)$	$a_v^k = \sum_{u\in\text{Parents}(v)} \alpha_{vu} \cdot h_u^{k-1}$	$h_v^k = \sigma(W^k \cdot [h_v^{k-1} \| a_v^k])$	Yes	Yes

$\mathbf{W_Q}, \mathbf{W_K}, \mathbf{W_V} \in \mathbb{R}^{d\times d_k}$. Transformers can be viewed as message-passing networks on fully connected graphs, disregarding the original graph structure. To incorporate structural biases, various positional encodings (PEs) are introduced. For instance, Graph Transformer [24] employs Laplacian eigenvectors as absolute PEs, while others use relative PEs based on graph kernels [1, 28] to modulate attention scores. The Structure-Aware Transformer (SAT) [2] reformulates self-attention as a kernel smoother, integrating local structural information through subgraph representations.

5 METHOD

5.1 Attention Based on Reachability

To address the scalability challenges of Transformers, which experience $O(|V|^2)$ complexity in memory and computation in MAT [36], a message-passing scheme is proposed that leverages attention based on reachability within a TAG. For each node v, $N_k(v)$, the set of nodes reachable from v within k steps, is computed, and messages are aggregated only from nodes within this set to minimize unnecessary calculations.

In a directed rooted tree $T = (V, E)$ with $|V|$ nodes, the runtime of TAG attention is $O(|V| \times k \times \Delta^+ \times d)$, where Δ^+ is the maximum out-degree of T, d is the feature dimension, and k is the number of steps considered in message passing. When $k = \infty$, the runtime becomes $O(|V| \times \ell \times \Delta^+ \times d)$, where ℓ is the depth of the tree T. In contrast, for a cyclic graph in the worst-case scenario, the runtime for computing the attention scores is:

$$\begin{aligned}\sum_{v\in V} |R(v)| \times d &= \sum_{v\in V} (|V| - 1) \times d \\ &= |V| \times (|V| - 1) \times d \\ &= O(|V|^2 \times d),\end{aligned} \tag{1}$$

where $R(v)$ represents the set of all other nodes in V (since every node can potentially reach every other node due to cycles).

In the proposed TAG attention mechanism, for each node v, attention over $N_k(v)$ was calculated, with a runtime upper bound of $O(|N_k(v)| \times d)$. Therefore, the total runtime of TAG attention is linear in the sum of the sizes of the receptive fields $\sum_{v\in V} |N_k(v)| \leq |V| \times k \times \Delta^+$ When $k = \infty$, it becomes $\sum_{v\in V} |N_k(v)| \leq |V| \times \ell \times \Delta^+$. This demonstrates that the TAG attention mechanism scales linearly with the number of nodes $|V|$ and mitigates the quadratic complexity in fully connected graphs [18].

5.2 Acyclic Coordination

To maintain acyclicity in the communication graph G_c, it is represented as a sequence of graphs across discrete time steps $t \in \{1, \ldots, T\}$. The attention weights in G_c, representing the influence of agent j on agent i at time t, are computed using an attention mechanism:

$$\text{Edge}_{ij}(t) = \frac{\exp\left(\kappa(x_i + PE_i,\ x_j + PE_j)/\sqrt{d_k}\right)}{\sum_{u\in N_k(i)} \exp\left(\kappa(x_i + PE_i,\ x_u + PE_u)/\sqrt{d_k}\right)}, \tag{2}$$

where:

- x_i and x_j are the feature vectors of agents i and j at time t.
- PE_i and PE_j are their positional encodings, reflecting hierarchical and directional positions.
- κ is a kernel function (e.g., scaled dot-product).
- d_k is the dimensionality of the key vectors.
- $N_k(i)$ is the set of nodes reachable from i within k steps.

Attention weights $\text{Edge}_{ij}(t)$ below a predefined threshold δ are set to zero to maintain sparsity and focus on significant interactions. The communication graph at each time step t is represented by a weighted adjacency matrix $M_t \in \mathbb{R}^{|V|\times|V|}$.

Inspired by [17, 40], acyclicity is enforced by projecting the communication graph onto an admissible solution set of graphs. Potential functions $\{p_t\}$ and corresponding skew-symmetric weight matrices $\{W_t\}$ are introduced, with $W_t = -W_t^\top$. The communication graph G_c can be reformulated as $v(W_t, p_t) = W_t \circ \text{ReLU}(\nabla p_t)$, where $\circ$ denotes the element-wise (Hadamard) product, and ∇p_t is the gradient of the potential function p_t. The potential function p_t is approximated by $p_t = -\Delta_{0,t}^\dagger \text{ div}\left(\frac{1}{2}(C(M_t) - C(M_t)^\top)\right)$, with $\Delta_{0,t}^\dagger$ being the pseudo-inverse of the graph Laplacian at time t, and div denoting the divergence operator. This formulation ensures acyclicity by enforcing a topological ordering where $p_t(j) > p_t(i)$ whenever there is a directed path from node i to node j.

The weighted adjacency matrix W_t is updated as:

$$[W_t]_{ij} = \begin{cases} 0, & \text{if } p_t(i) = p_t(j) \text{ or } M_t(i,j) = M_t(j,i) = 0; \\ \frac{M_t(i,j)}{p_t(j) - p_t(i)}, & \text{if } M_t(i,j) \neq 0 \text{ and } M_t(j,i) = 0; \\ \frac{M_t(j,i)}{p_t(j) - p_t(i)}, & \text{if } M_t(i,j) = 0 \text{ and } M_t(j,i) \neq 0. \end{cases} \tag{3}$$

This update ensures W_t maintains skew-symmetry and enforces acyclicity. The set $\mathcal{S}$ of valid skew-symmetric matrices is defined

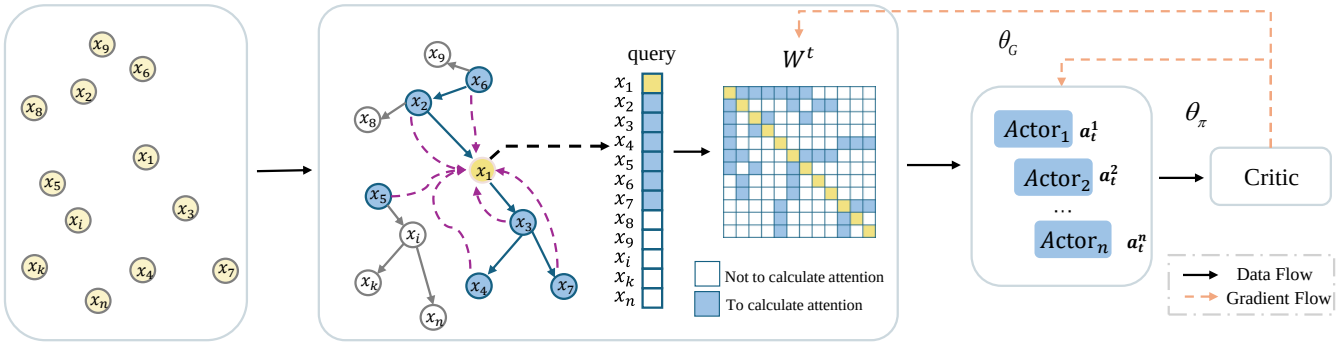


Figure 1: An overview of ACORN framework

by the constraints that the diagonal elements of W_t are zero and $(W_t)_{ij} = -(W_t)_{ji}$. To refine the adjacency matrices $\{W_t\}$, we optimize each W_t by solving $\hat{W}_t = \arg\min_{W_t \in \mathcal{S}} F(W_t \circ \text{ReLU}(\nabla p_t), X_t)$, where F is a loss function (e.g., negative critic output), and X_t represents additional variables at time t.

To optimize each temporal graph G_c, the objective is to maximize the critic's output $Q(s_t, G_t)$ while maintaining the skew-symmetry and acyclicity of W_t. The gradient update for the parameters θ is:

$$\nabla_\theta L(\theta) = \mathbb{E}\left[\nabla_{G_t} Q(s_t, G_t)\ \nabla_\theta v(W_t, p_t) + \gamma\ \nabla_\theta R(W_{t+1}, p_{t+1})\right], \tag{4}$$

where $R(W_{t+1}, p_{t+1})$ is a regularization term or reward, and γ is the discount factor. Only the elements in the upper triangular part of W_t are optimized to enforce skew-symmetry.

5.3 Auto-Regressive Policy Learning

The partial order inherent in the TAG emphasizes the importance of a node's predecessors and successors, making it a natural fit for an auto-regressive approach. In this setting, each agent's decision depends on preceding agents, effectively modeling the hierarchical and sequential relationships within the TAG as presented in Fig. 1. Formally, consider n agents with a permutation $\pi = \{i_1, i_2, \ldots, i_n\}$ over agent indices. The joint policy can be factorized in an auto-regressive manner based on the communication graph G_t:

$$\pi_\theta(a \mid \tau, C) = \prod_{m=1}^{n} \pi_\theta^{i_m}\left(a^{i_m} \mid \tau^{i_m}, C^{i_m}, a^{i_{1:m-1}}\right), \tag{5}$$

where:

- θ denotes the shared parameters of the agent networks.
- τ^{i_m} is the observation-action history of agent i_m.
- C^{i_m} represents the communication content for agent i_m, including information from predecessors.
- $a^{i_{1:m-1}}$ are the actions of preceding agents.

Specifically, the communication content C^{i_m} is defined as:

$$C^{i_m} = \sum_{j \in L(i_m)} W_t^{i_m j} h^j, \tag{6}$$

where $L(i_m)$ is the set of agents with directed edges to agent i_m in G_t, $W_t^{i_m j}$ is the weight from the communication matrix W_t, and h^j is the hidden state encoding of agent j's observation-action history τ^j. Each agent i_m takes actions considering the actions of previously executed agents $a^{i_1}, \ldots, a^{i_{m-1}}$. The policy aims to maximize the expected advantage given predecessors' actions.

At iteration $k+1$, given the permutation π, agent i_m optimizes its policy parameters $\theta_{k+1}^{i_m}$ by maximizing the expected advantage while ensuring stable updates, which can be formulated as:

$$\theta_{k+1}^{i_m} = \arg\max_{\theta^{i_m}} \mathbb{E}_{s\sim p_{\theta_k},\, a^{i_{1:m-1}}\sim\pi_{\theta_k}}\left[A_{\pi_{\theta_k}}^{i_m}\left(s,\ a^{i_{1:m-1}},\ a^{i_m}\right)\right], \tag{7}$$

subject to a constraint on policy update to prevent large deviations:

$$\mathbb{E}_{s\sim p_{\theta_k}}\left[D_{\text{KL}}\left(\pi_{\theta_{k+1}}^{i_m}(\cdot \mid s, a^{i_{1:m-1}}),\ \pi_{\theta_k}^{i_m}(\cdot \mid s, a^{i_{1:m-1}})\right)\right] \le \delta, \tag{8}$$

where δ is a threshold hyperparameter, and D_{KL} denotes the Kullback-Leibler divergence.

However, computing the KL-divergence constraint in Equation 8 can be intractable in practice. To make the optimization tractable, we adopt a clipped surrogate objective inspired by established policy optimization techniques [30]. The optimization becomes:

$$\theta_{k+1}^{i_m} = \arg\max_{\theta^{i_m}} \mathbb{E}_{s\sim p_{\theta_k},\, a^{i_{1:m-1}}\sim\pi_{\theta_k}}\Big[\min\Big(r^{i_m}(\theta)A_{\pi_{\theta_k}}^{i_m}\left(s,\ a^{i_{1:m}}\right),\ \text{clip}\left(r^{i_m}(\theta),\ 1-\epsilon,\ 1+\epsilon\right)A_{\pi_{\theta_k}}^{i_m}\left(s,\ a^{i_{1:m}}\right)\Big)\Big], \tag{9}$$

where:

$$r^{i_m}(\theta) = \frac{\pi_\theta^{i_m}\left(a^{i_m} \mid s,\ a^{i_{1:m-1}}\right)}{\pi_{\theta_k}^{i_m}\left(a^{i_m} \mid s,\ a^{i_{1:m-1}}\right)}, \tag{10}$$

and ϵ is a small positive constant controlling the clipping range.

5.4 Sequence-Based Critic Learning

A sequence-based critic learning method is proposed to complement auto-regressive policy learning. This approach ensures that the variance of the sequential advantage is upper bounded by the variance of the counterfactual advantage, thereby improving learning stability. In auto-regressive policy learning, each agent i_m follows a sequence to execute action a^{i_m}, considering predecessors' observation-action histories $\tau^{i_1}, \ldots, \tau^{i_{m-1}}$. The critic evaluates the

Q-value for the joint action $a = [a^{i_1}, a^{i_2}, \ldots, a^{i_n}]$ while simultaneously updating the communication graph G_t at each time step. The loss function for training the critic is:

$$\mathcal{L}_t(\phi) = \left(y_t^{(\lambda)} - Q_\phi(s_t, a_t, G_t)\right)^2, \tag{11}$$

where the target value $y_t^{(\lambda)}$ is computed using λ-returns:

$$y_t^{(\lambda)} = r_t + \gamma \left[\lambda y_{t+1}^{(\lambda)} + (1-\lambda) Q_{\phi^-}(s_{t+1}, a_{t+1}, G_{t+1})\right], \tag{12}$$

and ϕ^- denotes the parameters of the target critic network, periodically updated for stability. The critic evaluates each agent's action impact in sequence, ensuring the communication graph G_t evolves to support optimal coordination. The gradient update for critic ϕ is:

$$\nabla_\phi \mathcal{L}_t(\phi) = -2\left(y_t^{(\lambda)} - Q_\phi(s_t, a_t, G_t)\right) \nabla_\phi Q_\phi(s_t, a_t, G_t). \tag{13}$$

Additionally, the communication graph G_t is updated by optimizing its parameters to maximize the critic's output. The gradient update for parameters θ_G governing G_t is:

$$\nabla_{\theta_G} L(\theta_G) = \mathbb{E}\left[\nabla_{G_t} Q_\phi(s_t, a_t, G_t)\, \nabla_{\theta_G} G_t\right], \tag{14}$$

where θ_G represents parameters controlling the communication graph (e.g., in the attention mechanism). By integrating this sequence-based approach into critic learning, our framework ensures the communication graph G_t remains acyclic and evolves to reflect complex inter-agent dependencies, leading to improved coordination and policy convergence. The overall training and evaluation process is outlined in Algorithm 1.

Algorithm 1 Acyclic Coordination with Reachability Networks

1: **Initialize** policy parameters θ, critic parameters ϕ, and communication parameters θ_G.
2: **for** each iteration $k = 1, 2, \ldots$ **do**
3: **Collect** trajectories $\{\tau\}$ by executing the current policy π_{θ_k}.
4: **Update** communication graphs G_t and parameters θ_G using the attention mechanism (Equation 6).
5: **Compute** advantage estimates $A_{\pi_{\theta_k}}$ using the critic Q_ϕ.
6: **Update** policy parameters θ by optimizing the surrogate objective (Equation 9).
7: **Update** critic parameters ϕ by minimizing the loss $\mathcal{L}_t(\phi)$ (Equation 11).
8: **Periodically** update target critic parameters $\phi^- \leftarrow \phi$.
9: **end for**

6 EXPERIMENTS

6.1 Experimental Setup

To assess the effectiveness of ACORN's acyclic coordination, four common MARL benchmarks were utilized: SMAC [29], SMACV2 [7], Google Research Football [12], and the Multi-Agent MuJoCo benchmark [4]. These environments are designed to challenge agents' coordination and communication abilities, making them suitable for evaluating the effects of acyclic coordination. All the experiments are calculated on a computer with Ubuntu 18.04, two Intel(R) Xeon(R) Gold 6230R CPUs @2.10 GHz and 256 GB of RAM.

ACORN was compared with several SOTA MARL methods, including MAPPO [39] and QMIX [26], known for their stable policy updates and efficiency. HAPPO [41] is a trust-region learning method suitable for adaptive agents in dynamic environments. Additionally, A2PO [34] and MAT [36] were included, as they employ sequential policy updates but do not explicitly enforce acyclic communication structures. CASEC [33] drops edges in the communication graph based on payoff variance but does not guarantee acyclicity. The hyperparameters specified in the original papers of the baseline algorithms were adhered to for a fair comparison (details are provided in the *Appendix B.1*).

6.2 Comparisons with SOTA Methods

Figure 3 presents a comprehensive comparison between ACORN and baseline methods on the SMAC benchmark. The mean and standard deviation over five random seeds are reported to ensure reliable results. ACORN demonstrates superior performance in 6 out of 9 tasks in SMAC. Notably, in the more challenging hard+ maps, ACORN consistently outperforms other methods, illustrating the effectiveness of acyclic coordination in enhancing performance. Additional results can be found in *Appendix B.2*.

In SMACV2, which introduces increased stochasticity through random team compositions, start positions, and diverse unit types, ACORN shows substantial improvements across all seven tasks (Table 2). For example, in the *zerg_20vs20* task, ACORN achieves a win rate of 37.6%, which is 10.2% higher than QMIX and 27.2% higher than MAT. In the most challenging *zerg_20vs23* task, ACORN attains a win rate of 25.5%, surpassing MAT by 10.6% and QMIX by 15.4%. These significant gains highlight that ACORN provides a robust solution to overcome complex communication. Figure 2

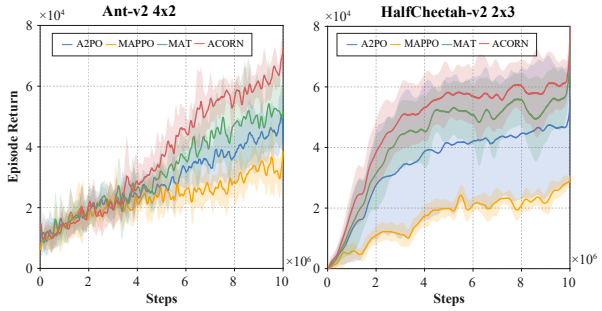


Figure 2: Performance evaluation of ACORN on MaMuJOCO. The mean and standard deviation over 5 random seeds are reported, compared with A2PO, MAT and MAPPO.

illustrates the performance of ACORN on the MaMuJoCo benchmark, a set of continuous control tasks involving multiple agents with heterogeneous action spaces. Compared to other methods such as A2PO, MAT, and the baseline method MAPPO—which demonstrate strong performance in these tasks—ACORN achieves an average performance improvement of 12%. This demonstrates its superior coordination capabilities in environments with multiple action spaces and diverse agent objectives.

In GRF, ACORN is evaluated on five common tasks, as shown in Table 3. ACORN outperforms baseline methods across all scenarios,

demonstrating its ability to efficiently manage communication and coordination even in complex, dynamic environments. For instance, in the academy counterattack hard (CA-hard) scenario—where four attackers face two defenders and a keeper, while all remaining players run back toward the ball—significant improvements are observed with ACORN. This scenario tests the algorithm's effectiveness in coordinating agents require precise timing and teamwork.

6.3 Comparisons in Usage of Wall Time

To evaluate the efficiency of acyclic coordination in terms of computational overhead, the wall-clock time usage of ACORN was compared with that of other methods on the Multi-Agent MuJoCo *Humanoid 9|8* task. ACORN's TAG reduces time complexity by minimizing redundant computations inherent in cyclic or fully connected communication graphs. As shown in Table 4, although ACORN requires slightly more training time due to the overhead of constructing the TAG, it achieves significantly higher rewards. After 1.5 hours of training, ACORN achieves a reward of 2505.48, which is approximately 60% higher than MAT and 304% higher than MAPPO. This indicates that the acyclic coordination allows ACORN to train more effectively, leading to better performance.

Similarly, in terms of environment steps, as shown in Table 5, ACORN achieves higher rewards with fewer steps. At 1×10^7 steps, ACORN attains a reward of 7462.39, which is approximately 21.4% higher than MAT and 238.9% higher than MAPPO. This demonstrates that acyclic coordination improves learning efficiency, allowing agents to learn more effectively from interactions.

6.4 Ablation Study

To evaluate the impact of acyclic coordination, an ablation study was conducted comparing ACORN with three variants: ACORN-SG (Static Graph), where agents communicate over a randomly generated static graph without enforcing acyclicity; ACORN-TG (Temporal Graph), which utilizes a temporal graph that may contain cycles; and ACORN-NG (No Graph), where no explicit communication graph is employed, allowing agents to act independently. Fig. 4 illustrates the performance of each variant across different tasks. In the *Terran* 20vs20 scenario, ACORN achieves a win rate approximately 15% higher than ACORN-SG and 20% higher than ACORN-NG. Additionally, ACORN-TG demonstrates intermediate performance, suggesting that while temporal graphs can mitigate some redundancy, they do not match the effectiveness of the acyclic structure provided by TAG. These findings indicate that enforcing acyclicity in the communication graph significantly enhances coordination and overall performance by reducing communication redundancy and facilitating clearer information flow among agents.

In addition to performance metrics, the performance of each variant is evaluated to assess computational efficiency. Table 6 presents the training time required at key environment steps. Specifically, ACORN-NG does not exhibit higher efficiency within the same training time, indicating that ACORN's reachability attention mechanism, despite the overhead of maintaining an acyclic graph, facilitates more effective communication and validates the combination of GNNs with Transformers. ACORN-RG shows significantly

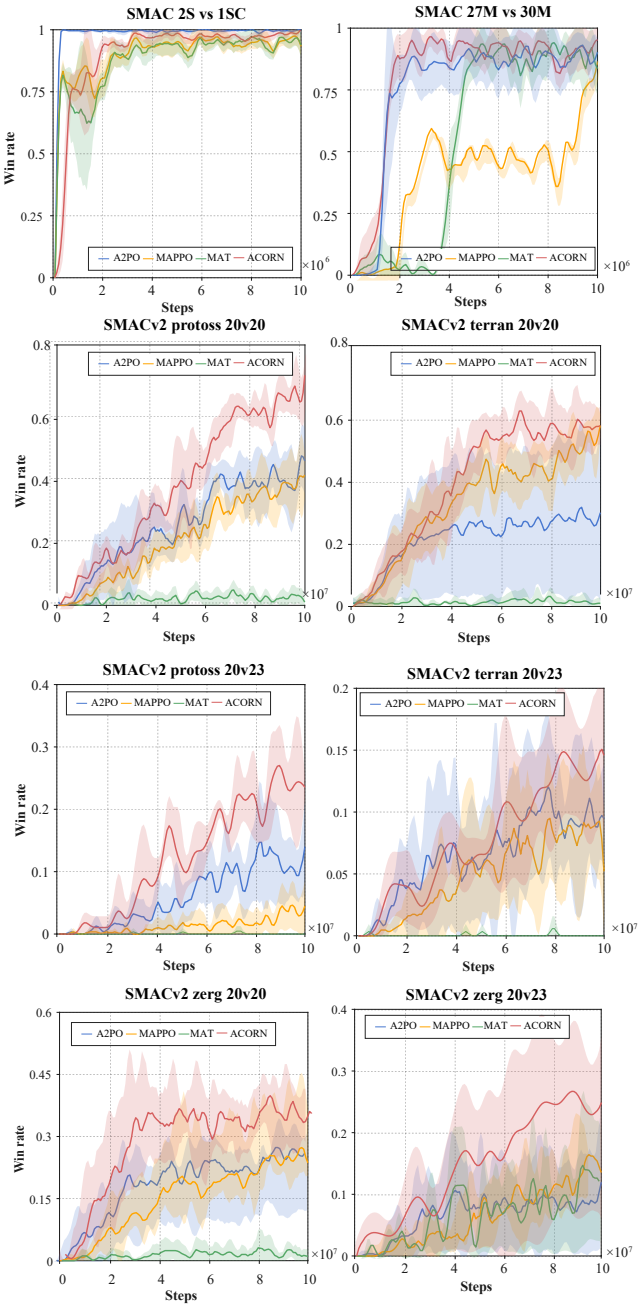


Figure 3: Performance evaluation of ACORN on SMAC and SMACV2 in hard+ maps. The mean and standard deviation over 5 random seeds are reported, compared with A2PO, MAT, CASEC, MAPPO, and QMIX.

reduced efficiency with higher variance, suggesting that predefined static graphs, while avoiding computational redundancy from dynamic graph modeling, fail to accurately capture communication relationships among agents. Similarly, ACORN-TG requires more training time than ACORN-RG but still does not achieve ACORN's

Table 2: Median win rates and standard deviations on SMAC and SMACV2 tasks. ACORN achieves higher win rates, demonstrating the effectiveness of acyclic coordination. Best performance is indicated by the bold face numbers.

Map	Difficulty	MAT	MAPPO	QMIX	CASEC	HAPPO	A2PO	ACORN	Steps
MMM	Easy	**100.0(2.2)**	96.9(0.6)	95.3(2.5)	91.7(5.2)	95.3(2.5)	**100.0(1.1)**	**100.0(0.6)**	1×10^7
3s_vs_5z	Hard	**100.0(1.9)**	96.9(1.9)	98.4(2.4)	96.2(22.9)	96.3(0.7)	**100.0(2.5)**	**100.0(1.7)**	1×10^7
8m_vs_9m	Hard	95.3(1.1)	96.9(0.6)	92.2(2.0)	92.2(4.4)	96.9(3.8)	**100.0(1.0)**	97.1(1.3)	1×10^7
10m_vs_11m	Hard	98.1(1.4)	93.8(18.7)	95.3(1.0)	87.5(9.7)	98.4(3.0)	97.9(0.5)	**100.0(0.5)**	1×10^7
6h_vs_8z	Hard+	95.8(1.3)	87.5(1.5)	9.4(2.0)	89.4(4.0)	87.5(1.5)	90.6(1.3)	**96.6(0.4)**	5×10^7
3s5z_vs_3s6z	Hard+	96.5(1.3)	84.4(34.0)	82.8(5.3)	93.6(0.4)	37.5(13.2)	93.8(19.8)	**98.2(0.7)**	5×10^7
MMM2	Hard+	96.8(2.6)	90.6(2.8)	87.5(2.6)	74.3(0.7)	51.6(9.0)	**98.4(1.2)**	97.9(2.5)	5×10^7
27m_vs_30m	Hard+	95.4(0.7)	93.8(3.8)	39.1(9.8)	0.6(0.8)	90.6(4.8)	88.7(14.5)	**98.2(1.7)**	5×10^7
corridor	Hard+	90.2(1.7)	**100.0(1.2)**	84.4(2.5)	82.8(3.1)	96.9(1.0)	**100.0(0)**	98.8(0.8)	5×10^7
SMACV2 Tasks									
protoss_5vs5	-	60.4(0.7)	56.2(3.2)	65.6(3.9)	42.6(2.6)	57.5(1.2)	62.3(1.2)	**71.4(2.5)**	1×10^8
terran_5vs5	-	61.2(1.4)	53.1(2.7)	62.5(3.8)	40.2(1.8)	57.5(1.3)	50.1(4.6)	**65.5(2.3)**	1×10^8
zerg_5vs5	-	55.3(2.1)	40.6(7.0)	34.4(2.2)	25.5(5.6)	42.5(2.5)	46.3(2.9)	**60.1(2.7)**	1×10^8
zerg_10vs10	-	48.4(1.7)	37.5(3.2)	40.6(3.4)	20.3(1.6)	28.4(2.2)	36.6(3.7)	**50.4(1.3)**	1×10^8
zerg_10vs11	-	24.2(5.7)	29.7(3.8)	25.0(3.9)	15.3(2.1)	16.2(0.6)	12.7(0.4)	**46.3(0.4)**	1×10^8
zerg_20vs20	-	10.7(5.5)	20.3(1.7)	27.4(3.6)	0.0(0.0)	7.1(2.6)	9.6(1.5)	**37.6(2.7)**	1×10^8
zerg_20vs23	-	14.9(6.7)	15.7(5.6)	10.1(5.4)	0.0(0.0)	0.0(0.0)	2.7(1.1)	**25.5(7.6)**	1×10^8

Table 3: Performance results in GRF across various scenarios

Scenarios	ACORN	MAT	QMIX	CASEC
PS	**96.93 (0.27)**	94.92 (0.85)	68.05 (5.58)	16.67(9.43)
RPS	**87.30 (1.78)**	76.83 (3.57)	48.61 (3.29)	27.20(6.27)
3v.1	**95.69 (6.31)**	88.03 (4.15)	57.24 (4.46)	18.12 (4.46)
CA (easy)	**91.87 (3.69)**	87.76 (6.40)	45.81 (13.68)	37.08 (7.28)
CA (hard)	**92.26 (4.88)**	77.38 (10.95)	71.19 (11.31)	52.14 (13.82)

Table 4: Performance comparison in training time (in hours). ACORN trains effectively to achieve higher rewards.

Algorithm	Training Time (Hours)		
	0.5	1	1.5
MAPPO	445.65 ± 11.43	555.34 ± 10.05	811.70 ± 384.30
CoPPO	463.18 ± 10.42	551.50 ± 12.35	613.08 ± 25.57
HAPPO	468.13 ± 62.80	573.01 ± 77.86	689.67 ± 118.64
A2PO	436.54 ± 10.71	636.14 ± 115.58	1884.11 ± 172.37
MAT	515.92 ± 17.10	764.26 ± 23.14	1562.27 ± 428.37
ACORN	525.46 ± 6.82	925.37 ± 54.96	2505.48 ± 272.42

superior performance. This indicates that maintaining acyclicity is crucial for sequence policy learning. These findings show that enforcing an acyclic communication structure can enhance coordination and performance, with a modest increase in training time.

6.5 Case Study

A case study was conducted on the *MMM2* scenario in SMAC to analyze how acyclic coordination impacts agent interactions during training, as shown in Fig. 5. In this scenario, agents must coordinate complex strategies involving different unit types. ACORN's

Table 5: Performance comparison at key environment steps. ACORN achieves higher rewards with efficient use of steps due to acyclic coordination.

Algorithm	Training Steps		
	1×10^6	4×10^6	1×10^7
MAPPO	494.36 ± 12.96	561.24 ± 14.82	2199.93 ± 1700.22
CoPPO	522.24 ± 13.13	562.80 ± 12.49	676.57 ± 42.25
HAPPO	550.75 ± 69.06	616.11 ± 93.12	1610.38 ± 826.31
A2PO	538.77 ± 39.45	761.37 ± 197.91	5595.27 ± 478.50
MAT	472.60 ± 29.40	2147.80 ± 157.90	6147.80 ± 276.40
ACORN	574.62 ± 17.68	1274.62 ± 29.83	7462.39 ± 169.70

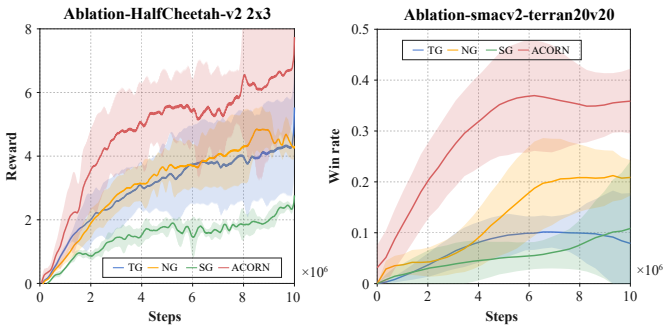


Figure 4: Ablation study on different communication graph structures. ACORN consistently outperforms other variants, highlighting the importance of acyclic coordination.

TAG constructs an acyclic communication graph that dynamically

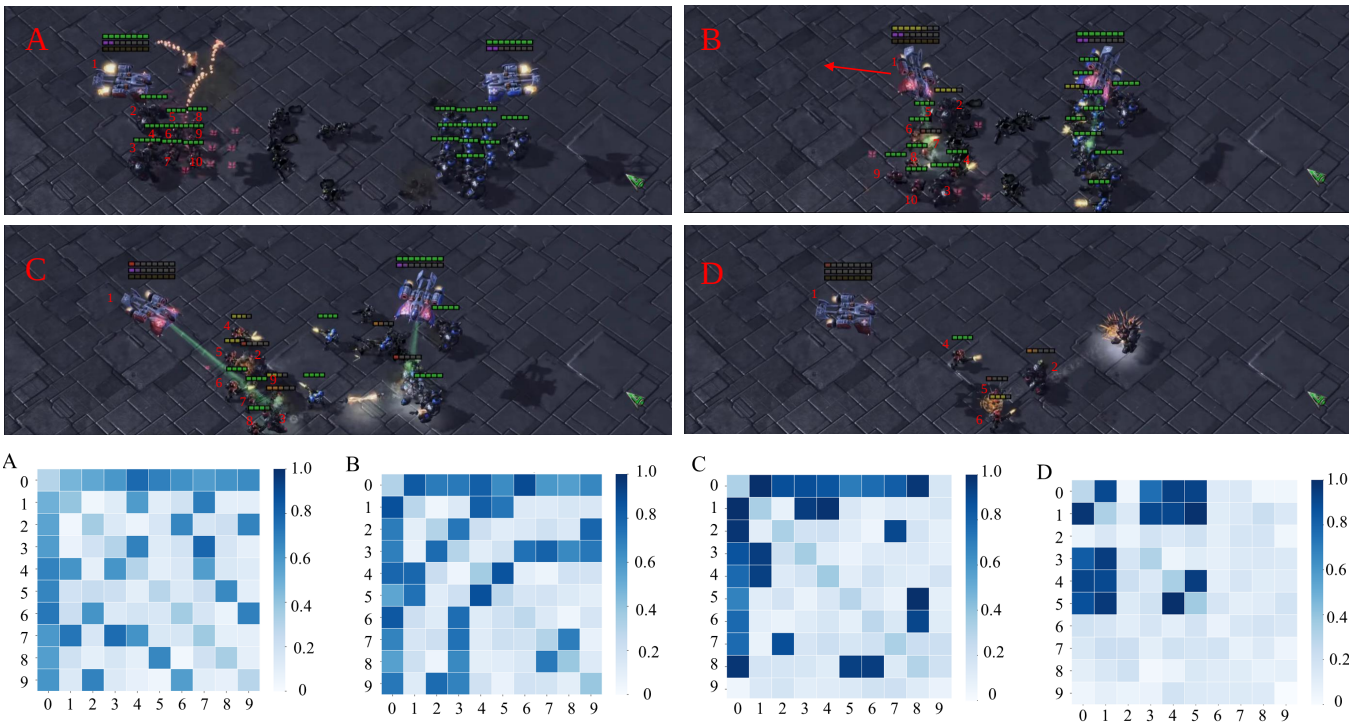


Figure 5: Case study on SMAC MMM2. The communication graph dynamically evolves, maintaining acyclicity and reflecting changes in the environment. This adaptation enhances agent coordination and performance.

Table 6: Training time comparison at key environment steps in HalfCheetah-v2 2x3.

Algorithm	Training Steps		
	1×10^6	4×10^6	1×10^7
ACORN-TG	1477.32 ± 201.43	3424.97 ± 962.56	6569.51 ± 847.83
ACORN-SG	1272.45 ± 371.96	2813.61 ± 1142.73	4331.94 ± 742.91
ACORN-NG	1626.75 ± 524.49	4120.43 ± 1493.74	6825.02 ± 1116.35
ACORN	2107.32 ± 205.03	5231.75 ± 1018.65	8141.93 ± 741.05

evolves with environmental changes. This ensures that agents receive relevant information without cycles, preventing feedback loops that can confuse decision-making processes. The dynamic adjustment of the communication graph leads to more effective coordination strategies. Compared to MAT, which employs a fully connected communication structure, ACORN reduces communication redundancy by approximately 30%, as measured by the number of redundant messages exchanged. This reduction contributes to more efficient communication and improved overall performance. A more detailed comparison of agent communication in the MMM2 scenario between MAT and ACORN is provided in *Appendix B.3.*

7 CONCLUSION

Efficient communication and coordination are crucial in multi-agent reinforcement learning (MARL) due to the complex interdependencies among agents. ACORN, a novel framework, is introduced to leverage acyclic coordination through TAG and a reachability-based attention mechanism, mitigating communication redundancy and reducing computational overhead. By selectively updating important nodes based on high GNN graph weights, ACORN significantly lowers time complexity from $(O(|V|^2 d))$ to $(O(|V| \times nk \times d))$, enhancing computational efficiency. Experiments on benchmarks such as SMAC, SMACV2, and Multi-Agent MuJoCo demonstrate that ACORN consistently outperforms state-of-the-art methods. Specifically, in challenging SMACV2 scenarios, an average improvement of 11% over MAT and a 17% improvement within the same training time and steps are achieved. These results validate the effectiveness of enforcing acyclic communication structures and selective attention in enhancing both performance and computational efficiency in MARL. Although ACORN enhances communication efficiency in multi-agent systems, the construction of TAG introduces additional computations that cannot be performed in real time. This limitation is likely to affect scalability when faced with a large number of agents or rapidly changing interactions. Further optimization of TAG construction is required in future work.

ACKNOWLEDGMENTS

This study was supported by the (1) Shanghai Engineering Research Center of AI & Robotics, Fudan University, China, and (2) Engineering Research Center of AI & Robotics, Ministry of Education, China. This study was partially supported by Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103).

REFERENCES

[1] Deng Cai and Wai Lam. 2020. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 7464–7471.

[2] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. 2022. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*. PMLR, 3469–3489.

[3] Jinchao Chen, Tingyang Li, Ying Zhang, Tao You, Yantao Lu, Prayag Tiwari, and Neeraj Kumar. 2023. Global-and-local attention-based reinforcement learning for cooperative behaviour control of multiple UAVs. *IEEE Transactions on Vehicular Technology* (2023).

[4] Christian Schroeder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2020. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *arXiv preprint arXiv:2003.06709* 19 (2020).

[5] Shifei Ding, Wei Du, Ling Ding, Jian Zhang, Lili Guo, and Bo An. 2023. Multiagent reinforcement learning with graphical mutual information maximization. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[6] Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. 2021. Learning correlated communication topology in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 456–464.

[7] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. 2022. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489* (2022).

[8] Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. 2022. Revisiting some common practices in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2206.07505* (2022).

[9] Matteo Gallici, Mario Martin, and Ivan Masmitja. 2023. TransfQMix: Transformers for Leveraging the Graph Structure of Multi-Agent Reinforcement Learning Problems. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 1679–1687.

[10] Songyang Han, Shanglin Zhou, Jiangwei Wang, Lynn Pepin, Caiwen Ding, Jie Fu, and Fei Miao. 2023. A multi-agent reinforcement learning approach for safe and efficient behavior planning of connected autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems* (2023).

[11] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. 2021. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*.

[12] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. 2020. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 4501–4510.

[13] Hepeng Li and Haibo He. 2023. Multiagent trust region policy optimization. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[14] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. 2020. Deep implicit coordination graphs for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.11438* (2020).

[15] Yihong Li, Xiaoxi Zhang, Tianyu Zeng, Jingpu Duan, Chuan Wu, Di Wu, and Xu Chen. 2023. Task placement and resource allocation for edge machine learning: A gnn-based multi-agent reinforcement learning paradigm. *IEEE Transactions on Parallel and Distributed Systems* (2023).

[16] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*. Elsevier, 157–163.

[17] Zeyang Liu, Lipeng Wan, Xue Sui, Zhuoran Chen, Kewu Sun, and Xuguang Lan. 2023. Deep hierarchical communication graph in multi-agent reinforcement learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 208–216.

[18] Yuankai Luo, Veronika Thost, and Lei Shi. 2023. Transformers over Directed Acyclic Graphs. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[19] Zefang Lv, Liang Xiao, Yifan Chen, Haoyu Chen, and Xiangyang Ji. 2024. Safe Multi-Agent Reinforcement Learning for Wireless Applications Against Adversarial Communications. *IEEE Transactions on Information Forensics and Security* (2024).

[20] Afshin Oroojlooy and Davood Hajinezhad. 2023. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence* 53, 11 (2023), 13677–13722.

[21] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems* 34 (2021), 12208–12221.

[22] Kexing Peng, Tinghuai Ma, Li Jia, and Huan Rong. 2024. Enhancing Collaboration in Heterogeneous Multiagent Systems Through Communication Complementary Graph. *IEEE Transactions on Cybernetics* (2024).

[23] Zhiqiang Pu, Huimu Wang, Zhen Liu, Jianqiang Yi, and Shiguang Wu. 2022. Attention enhanced reinforcement learning for multi agent cooperation. *IEEE Transactions on Neural Networks and Learning Systems* 34, 11 (2022), 8235–8249.

[24] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems* 35 (2022), 14501–14515.

[25] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems* 33 (2020), 10199–10210.

[26] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research* 21, 1 (2020), 7234–7284.

[27] Yan Ren, Heng Zhang, Linkang Du, Zhikun Zhang, Jian Zhang, and Hongran Li. 2024. Stealthy Black-Box Attack With Dynamic Threshold Against MARL-Based Traffic Signal Control System. *IEEE Transactions on Industrial Informatics* (2024).

[28] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-supervised graph transformer on large-scale molecular data. *Advances in neural information processing systems* 33 (2020), 12559–12571.

[29] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 2186–2188.

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[31] Huaze Tang, Hengxi Zhang, Zhenpeng Shi, Xinlei Chen, Wenbo Ding, and Xiao-Ping Zhang. 2023. Autonomous Swarm Robot Coordination via Mean-Field Control Embedding Multi-Agent Reinforcement Learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8820–8826.

[32] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[33] Tonghan Wang, Liang Zeng, Weijun Dong, Qianlan Yang, Yang Yu, and Chongjie Zhang. 2021. Context-aware sparse deep coordination graphs. *arXiv preprint arXiv:2106.02886* (2021).

[34] Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. 2022. Order Matters: Agent-by-agent Policy Optimization. In *The Eleventh International Conference on Learning Representations*.

[35] Yajie Wang, Dianxi Shi, Chao Xue, Hao Jiang, Gongju Wang, and Peng Gong. 2020. AHAC: actor hierarchical attention critic for multi-agent reinforcement learning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 3013–3020.

[36] Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. 2022. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems* 35 (2022), 16509–16521.

[37] Ming Yang, Kaiyan Zhao, Yiming Wang, Renzhi Dong, Yali Du, Furui Liu, Mingliang Zhou, and Leong Hou U. 2024. Team-wise effective communication in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 38, 2 (2024), 36.

[38] Qianlan Yang, Weijun Dong, Zhizhou Ren, Jianhao Wang, Tonghan Wang, and Chongjie Zhang. 2022. Self-organized polynomial-time coordination graphs. In *International Conference on Machine Learning*. PMLR, 24963–24979.

[39] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems* 35 (2022), 24611–24624.

[40] Yue Yu, Tian Gao, Naiyu Yin, and Qiang Ji. 2021. DAGs with no curl: An efficient DAG structure learning approach. In *International Conference on Machine Learning*. PMLR, 12156–12166.

[41] Yifan Zhong, Jakub Grudzien Kuba, Siyi Hu, Jiaming Ji, and Yaodong Yang. 2023. Heterogeneous-Agent Reinforcement Learning. *arXiv preprint arXiv:2304.09870* (2023).

[42] Ziqing Zhou, Chun Ouyang, Linqiang Hu, Yi Xie, Yuning Chen, and Zhongxue Gan. 2024. A framework for dynamical distributed flocking control in dense environments. *Expert Systems with Applications* 241 (2024), 122694.