applied sciences

MDPI

Article

Group Attention Aware Coordination Graph

Ziyan Fang, Wei Liu and Yu Zhang *

College of Intelligence Science and Technology, National University of Defence Technology, Changsha 410073, China; fangziyan0531@163.com (Z.F.); 19159081836@163.com (W.L.)
* Correspondence: redarmy_zy@nudt.edu.cn

Abstract

Cooperative Multi-Agent Reinforcement Learning (MARL) relies on effective coordination among agents to maximize team performance in complex environments. However, existing coordination graph-based approaches often overlook dynamic group structures and struggle to accurately capture fine-grained inter-agent dependencies. In this paper, we introduce a novel method called the Group Attention Aware Coordination Graph (G2ACG), which builds upon the group modeling capabilities of the Group-Aware Coordination Graph (GACG). G2ACG incorporates a dynamic attention mechanism to dynamically compute edge weights in the coordination graph, enabling a more flexible and fine-grained representation of agent interactions. These learned edge weights guide a Graph Attention Network (GAT) to perform message passing and representation learning, and the resulting features are integrated into a global policy via QMIX for cooperative decision-making. Experimental results on the StarCraft II Multi-Agent Challenge (SMAC) benchmark show that G2ACG consistently outperforms strong baselines, including QMIX, DICG, and GACG, across various scenarios with diverse agent types and population sizes. Ablation studies further confirm the effectiveness of the proposed attention mechanism, demonstrating that both the number of attention heads and the number of GAT layers significantly affect performance, with a two-layer GAT and multi-head attention configuration yielding the best results.

Keywords: multi-agent reinforcement learning; coordination graph; group modeling; dynamic attention

Academic Editors: Thomas Lindner and Yutaka Ishibashi

Received: 20 July 2025
Revised: 11 September 2025
Accepted: 11 September 2025
Published: 24 September 2025

Citation: Fang, Z.; Liu, W.; Zhang, Y. Group Attention Aware Coordination Graph. *Appl. Sci.* **2025**, *15*, 10355. https://doi.org/10.3390/app151910355

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

Cooperative Multi-Agent Reinforcement Learning (MARL) aims to enable a group of agents to collaboratively solve complex tasks in a shared environment through collaboration and coordination [1]. Compared to single-agent reinforcement learning, the introduction of multiple agents brings additional challenges, particularly when dealing with vast state and action spaces, where traditional reinforcement learning algorithms prove inefficient [2]. A straightforward approach is to decompose the global training objective into individual tasks for each agent, solving for the maximization of the overall return by maximizing the action–value function of each single agent, as seen in algorithms like VDN [3] and QMIX [4]. However, real-world tasks are often hard to decompose into simple individual subtasks. This limitation motivates the need to model inter-agent relationships explicitly, often represented by coordination graphs (CGs), where nodes denote agents and edges capture their dependencies.

Recent advances in MARL have explored coordination graphs to model agent interactions, leading to three main branches: directly linking all nodes to create complete

graphs [2], weighting edges of fully connected graphs with attention mechanisms [5], and introducing group relationships to construct sparse graphs [2,6]. However, these algorithms fail to effectively balance capturing group dynamics and the accurate acquisition of weights between agents, resulting in sub-par performance. Moreover, they often overlook the need to adaptively model fine-grained inter-agent differences in complex environments. Take StarCraft II (StarCraft and StarCraft II are trademarks of Blizzard Enterainment) games as an example. Some maps, such as 2 Stalkers & 3 Zealots, feature diverse types of agents. This demands not only the efficient division of labor between their groups but also the real-time and precise construction of weight relationships between nodes.

Motivated by these limitations, we aim to enhance coordination graph expressiveness by introducing dynamic edge reasoning while preserving group structure modeling. To address the limitation of the current GACG algorithm, where the sampled graph lacks subsequent edge weight computation, resulting in weaker capability to capture node differences, we propose a novel improved algorithm: **Group Attention Aware Coordination Graph (G2ACG)**. This algorithm builds upon the GACG framework while preserving its group relationship modeling capabilities, and introduces one of the latest dynamic attention mechanisms [7] to enable the more precise capture of inter-agent relationships. Specifically, it integrates a multi-head attention module to dynamically generate edge weights, which are then used in a Graph Attention Networks (GATs) for message propagation and policy learning. The integration of these components significantly enhances multi-agent coordination efficiency in more complex environments. Recent studies have introduced various attention mechanisms to enhance multi-agent learning performance, such as the DAGMIX [8] and GA-Comm [9] algorithms. However, these methods still face challenges in complex, heterogeneous scenarios.

In order to validate the superiority of the G2ACG algorithm, we conducted tests on decentralized micromanagement tasks in the StarCraft II Multi-Agent Challenge (SMAC) environment [10]. Through experimental evaluation, we not only verified the algorithm's superior performance but also investigated the contributions of its individual components.

The contributions of this paper are summarized as follows:

- We provide a systematic categorization of existing coordination graph-based MARL algorithms, analyzing their construction paradigms and limitations to guide future research.
- Building upon the GACG framework, we integrate its group relationship modeling while introducing a novel dynamic attention mechanism that breaks the weight-sharing constraint in traditional GAT networks [11]. This design enables agents to flexibly assign attention weights to different neighbors, enhancing the model's expressiveness in representing diverse inter-agent interactions.
- Through conducting experiments on the StarCraft II (SMAC) environment [10], we demonstrate that our proposed G2ACG algorithm outperforms other baseline methods, showcasing the advantages of combining group relationships with dynamic attention mechanisms. Additionally, ablation studies validate the effectiveness of key components, highlighting the importance of multi-head attention and GAT layer depth.

2. Related Work

In the field of MARL, coordination graph-based approaches provide effective solutions for enabling cooperation among agents [5,12]. These methods aim to represent agent interactions as structured graphs, allowing information exchange, policy decomposition, and coordination to be more interpretable and efficient. To adapt to environmental changes and task requirements, current algorithms need to monitor environmental dynamics and agent states in real time and dynamically adjust coordination relationships [13]. The following are some major algorithms and their characteristics.

DCG [5] is one of the earliest proposed algorithms. It factors the joint value function according to a coordination graph into pairwise payoffs, and employs parameter sharing and low-rank approximations to manage complexity. However, its static graph structure limits its ability to adapt to changing agent behaviors. DICG [14] uses attention mechanisms and attempts to dynamically learn coordination relationships among agents through a deep network structure. GCS [15] proposes decomposing the joint team policy into a graph generator and a graph-based coordination policy, enabling the graph structure to evolve during training. GACG [2] leverages group-level behavior similarities and represents edges in the coordination graph as Gaussian distributions, combining agent–pair interaction and group-level dependencies. LTS-CG [16] calculates a probability matrix for agent pairs based on historical observation data and samples sparse coordination graphs for efficient information propagation. DMCG [6] extends traditional coordination graphs by introducing dynamic edge types to capture higher-order and indirect relationships among agents. GA-Comm [9] proposes a two-stage attention mechanism for multi-agent game abstraction, using hard attention to prune edges and soft attention to weight interactions. Co-GNN [17] treats nodes as agents that dynamically choose actions, enabling adaptive graph rewiring via jointly trained policy and environment networks. The corresponding algorithm can be found in Table 1.

While these approaches demonstrate promising progress in coordination modeling, they also exhibit key limitations. Many existing algorithms either overlook higher-order group structures or fail to capture fine-grained inter-agent variations in edge weights, which are essential for dynamic coordination. Consequently, they struggle to complete the adjustment and optimization of coordination relationships within short timeframes [18]. This limitation affects the overall performance and adaptability of multi-agent systems and results in insufficient capability when facing complex and uncertain environments.

Table 1. Comparison of coordination graph-based methods.

Approach	Core Methodology	Graph Type
DCG	Full connectivity of all nodes	Complete graph
DICG	Attention-weighted edges	Dynamic dense graph
GCS	Two-stage policy	Sparse graph
GACG	Group-wise Gaussian edges	Hierarchical graph
LTS-CG	History-based edge pruning	Sparse temporal graph
DMCG	Typed edge modeling	Multi-relational graph
GA-Comm	Two-stage attention	Dynamic sparse graph
Co-GNN	Node-level action selection	Adaptive directed graph

3. Background

We consider a classical Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [19], described with a tuple $\langle N, S, \{A_i\}_{i=1}^n, P, \{O_i\}_{i=1}^n, \{\pi_i\}_{i=1}^n, R, \gamma \rangle$. Let $N = \{N_1, N_2, \ldots, N_n\}$ denote the set of n agents. The true state of the environment is represented by $s \in S$. At each time step t, agent N_i receives an observation $o_i^t \in O^t$ and takes an action $\mathbf{a}_i^t \in A^t$ according to its policy π_i. The environment then transitions to the next state s' based on the state transition function $P(s'|s, \mathbf{a})$. All agents share a common reward function $R(s, \mathbf{a})$, and $\gamma \in [0, 1]$ is the discount factor. In a T-length scenario, the agents collaborate to maximize the joint action–value function at moment t:

$$Q_{\text{tot}}(s^t, \mathbf{a}^t) = \mathbb{E}_{s^{t+r}, \mathbf{a}^{t+r}} \left[\sum_{i=0}^{T} \gamma^i R(s^{t+i}, \mathbf{a}^{t+i}) \mid s^t, \mathbf{a}^t \right] \tag{1}$$

4. Methods

We adopt the GACG framework to generate a coordination graph. By incorporating a dynamic attention mechanism, we design the G2ACG algorithm, as illustrated in Figure 1. The algorithm can be broadly divided into three components:

- Group-Aware Coordination Graph Module (Blue Box). This module employs the group relationship capturing method from GACG [2]. It utilizes group relationships to derive a Gaussian distribution, which is then used to sample the edges of the coordination graph.
- Graph Convolution Module (Green Box). We introduce a dynamic attention mechanism [7] to dynamically compute the attention weights of nodes in the sampled coordination graph. Through multi-layer GCN (Graph Convolutional Network) [20] convolutions, the features of node messages in the coordination graph are extracted.
- Action Decision Module (Purple Box). Each group of agents determines the optimal Q-values based on their current observations and action inputs. These Q-values are then aggregated using the QMIX network [4] to derive the optimal value of the joint action–value function.

Below is an introduction to its implementation principles.

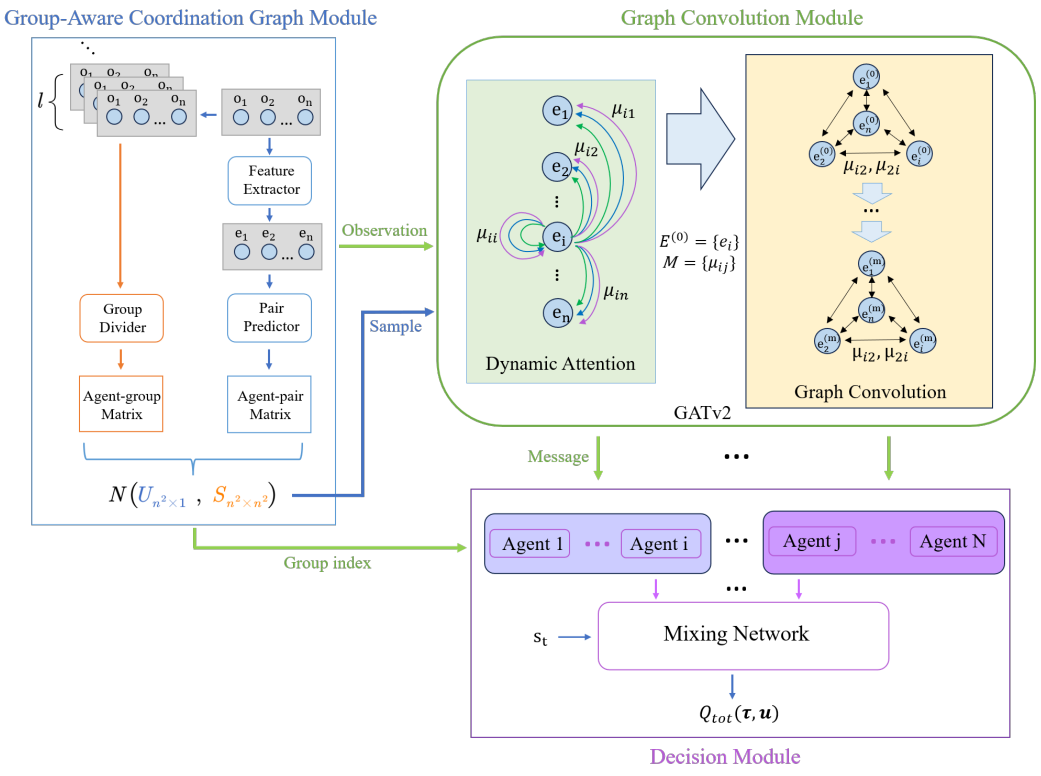

Figure 1. The framework of G2ACG, consisting of three main components: a Group-Aware Coordination Graph module (blue), a Graph Convolution Module (green), and an action decision module (purple). We use GATv2 network to generate the weight of edges dynamically.

4.1. Group-Aware CG

Our approach adopts the group relationship capturing method from GACG. For the observations of agents at time step t: $\{o_1^t, o_2^t, \ldots, o_n^t\}$, a feature extractor $f_{\text{ext}}(\cdot)$ extracts their hidden features $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n\}$, which is realized via a multi-layer perceptron (MLP). Then, pair predictor $f_{\text{pre}}(\cdot)$, an attention network, computes the weights between agent nodes

at the current timestep. The weight relations can be represented as an adjacency matrix, which is denoted as the agent–pair matrix:

$$\boldsymbol{e}_i = f_{\text{ext}}(\boldsymbol{o}_i) \tag{2}$$

$$u_{ij} = f_{\text{pre}}(\boldsymbol{e}_i, \boldsymbol{e}_j) \tag{3}$$

To facilitate the representation of edge weights, the matrix is reshaped into a column vector:

$$\boldsymbol{U}_{ij} = \text{vec}(u_{ij}) \tag{4}$$

On the other hand, we record observation trajectories over a fixed window length l (set to 10 in this experiment). Under the premise that agents with similar observations within a specific time period are likely to exhibit similar behaviors [2], the group divider partitions the agents into m distinct groups, yielding group relationships:

$$G = f_{\text{div}}(\boldsymbol{O}^{t-l:t}) \tag{5}$$

Here, $f_g(\cdot)$ maps the set of agents N to the set of groups G, and $\boldsymbol{O}^{t-l:t}$ represents the observation trajectories of agents from timestep $t-l$ to t. Based on the agent grouping, the agent-group matrix $\boldsymbol{S}$ is constructed:

$$s_{ij}^t = \begin{cases} 1 & a_i, a_j \in g_m \text{ at moment } t \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

This matrix indicates whether agents a_i and a_j belong to the same group at timestep t. The group relationships between agents can then be transformed into group relationships between edges in the coordination graph using the following method:

$$S_{L_1L_2}^t = \text{vec}(s_{ab}^t) \cdot \text{vec}(s_{cd}^t)^T \tag{7}$$

$S_{L_1L_2}^t = 1$ signifies that the four nodes a, b, c, and d on edges L_1 and L_2 are all in the same group at the current timestep; otherwise, the value is 0.

The weights between agents reflect their connectivity, while the group relationships reflect their dispersion [2]. Using these two matrices, the relationships between edges in the coordination graph can be represented as a two-dimensional Gaussian distribution, i.e.,

$$(\boldsymbol{e}_{ab}, \boldsymbol{e}_{cd}) \sim \mathcal{N}\left(\left(U_{ab}, U_{cd}\right), \begin{bmatrix} S_{L_1L_1} & S_{L_1L_2} \\ S_{L_2L_1} & S_{L_2L_2} \end{bmatrix} \right) \tag{8}$$

4.2. Group Attention Aware Cooperative for MARL

Utilizing the Gaussian distribution defined in Equation (8), we sample the edges of the Group-Aware Coordination Graph at each timestep, reshaping them into matrix form $\boldsymbol{C}^t$.

Definition 1 (Dynamic attention)**.** *For a query node set Q and key node set K, if there exists a family of scoring functions $f \in \mathcal{F}$, such that, for every query $i \in [v]$ and non-target key $j_{\neq\varphi(i)} \in [w]$, it satisfies*

$$f(q_i, k_{\varphi(i)}) > f(q_i, k_j) \tag{9}$$

then f is called a dynamic attention function, where $\varphi : [v] \to [w]$. This mechanism effectively addresses the limitation of static attention coefficients where the sorting is independent of the query nodes [7].

For the sampled sparse graph, we employ a dynamic attention mechanism [7] to compute weights between nodes. Compared to the standard Graph Attention Network (GAT) [11], GATv2 modifies the internal operation sequence: it applies the $\boldsymbol{a}$ layer (a single-layer feedforward network) after the nonlinearity $\boldsymbol{LeakyReLU}$ (with negative input slope 0.2), and applies the $\boldsymbol{W}$ layer after feature concatenation [7]. This yields the attention coefficients

$$z_{ij} = \boldsymbol{a}^T \boldsymbol{LeakyReLU}\left(\boldsymbol{W} \cdot [\boldsymbol{e}_i \| \boldsymbol{e}_j]\right) \tag{10}$$

where $\boldsymbol{e} = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n\}$ denotes features extracted in the preceding stage, $\boldsymbol{a}$ is the weight vector of the $\boldsymbol{a}$ layer, and $\boldsymbol{W}$ is the weight matrix. Cross-node normalization is achieved via softmax to obtain weights between feature nodes:

$$\mu_{ij} = \text{softmax}_j(z_{ij}) = \frac{\exp(z_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(z_{ik})} \tag{11}$$

The dynamic attention values μ_{ij} are integrated into the Graph Convolutional Network (GCN) [20] to generate the final message. The features $E^{(0)} = \{e_i\}$ serve as the original input, and the features of the l-th layer are obtained through the following iteration:

$$E^{(l)} = \text{ReLU}\left(\tilde{D}^{-\frac{1}{2}} M \tilde{D}^{-\frac{1}{2}} E^{(l-1)} W_c^{(l-1)}\right) \tag{12}$$

where $\tilde{D}_{ii} = \sum_j \mu_{ij}$, and $W_c^{(l)}$ is the trainable weight matrix of the l-th layer. The features $E^{(m)}$ obtained from the final convolution serve as the messages $\{m^t\}$, which, together with the agent's observation $\{o_i^t\}$ and the previous action $\{a_i^{t-1}\}$, are fed into the QMIX network. The mixing network then outputs the global optimal Q-value, based on which the next action a^t is determined.

5. Experiments and Results

In this section, we design experiments to investigate the following: (1) How does G2ACG perform in complex cooperative multi-agent tasks compared to other CG-based methods? (2) How do different numbers of GCN layers affect G2ACG's performance? (3) How does increasing the number of dynamic attention heads impact the final results? For the experiments in this study, we test our algorithm on decentralized micromanagement problems in the SMAC environment. We consider combat scenarios where two identical groups of units are symmetrically positioned on the map, with each scenario containing at least five agents. The environment difficulty level is set to 7. To ensure the robustness and reliability of our experimental methodology, all tests are conducted with five random seeds.

5.1. Compared with Other CG-Based Methods

We use QMIX [4], DICG [14], and GACG [2] as baseline algorithms for comparison (the characteristics of each method are summarized in Table 2):

- QMIX decomposes the global value function into a monotonic mixing of individual value functions through monotonicity constraints. It is suitable for multi-agent tasks in simple situations.
- DICG dynamically infers the coordination graph structure through a self-attention mechanism, enabling the method to be applied to more abstract multi-agent domains.
- GACG introduces higher-order group relationships and encourages behavioral specialization between groups through group-aware coordination, making it suitable for scenarios that emphasize group collaboration among diverse types of agents.

Table 2. Comparison of characteristics among different baseline algorithms (h denotes the number of attention heads, and l is the trajectory length).

Approach	Graph	Attention	Group	GC Time Complexity
QMIX	×	×	×	×
DICG	✓	Static	×	$O(hN^2)$
GACG	✓	×	✓	$O(lN^2)$
G2ACG	✓	Dynamic	✓	$O(hlN^2)$

✓indicates that the algorithm incorporates the structure/component, whereas × means it does not.

We conducted comparative experiments in four different map environments: 2s3z, 3s5z, 8m, and 8m_vs_9m. The experimental results are shown in Figure 2 (for intuitive visualization, all data in the results were smoothed with a coefficient of 0.6). Among them, G2ACG demonstrated optimal performance across all scenarios. In the 8m and 8m_vs_9m maps, G2ACG not only achieved significantly higher final win rates than the other three baseline algorithms, but also showed noticeably faster convergence speed. In the 2s3z and 3s5z maps, although the convergence speed was not significantly faster than QMIX and GACG, G2ACG still maintained higher final win rates than the other three baseline algorithms.

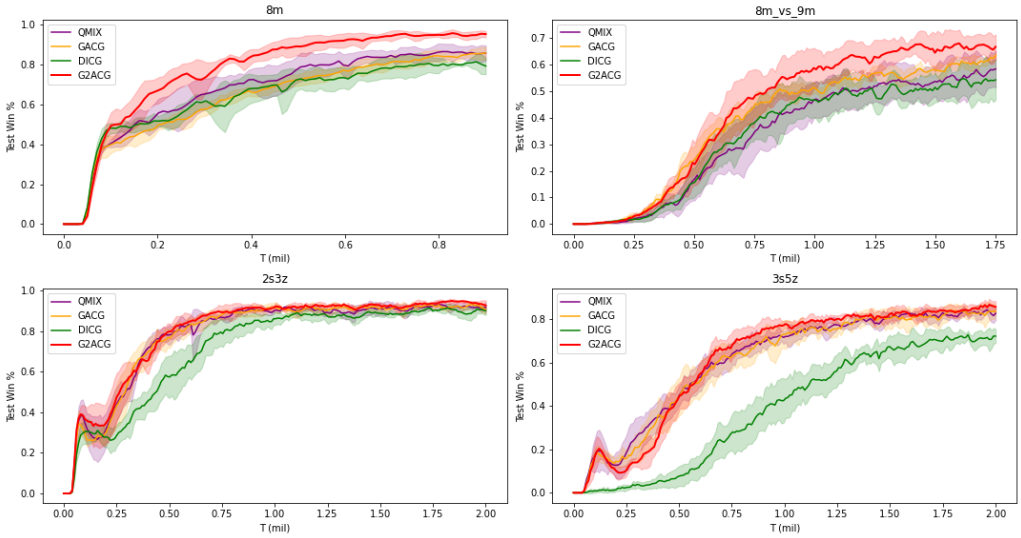

Figure 2. Performance of G2ACG and baseline algorithms on different SMAC map types. The x-axis represents the time steps (in millions), while the y-axis quantifies the test win rate in the games.

The three baseline algorithms each exhibited their respective limitations. The QMIX algorithm, lacking graph structure utilization, demonstrated poor adaptability in maps with larger numbers of agents such as 8m_vs_9m. DICG, which neglects the importance of group relationships, showed significantly weaker learning performance in maps like 2s3z and 3s5z that require clear inter-group specialization among diverse agent types. As for GACG, its failure to incorporate attention mechanisms resulted in inflexible weight calculations between agents, leading to substantially lower win rates compared to G2ACG in maps with numerous agents such as 8m and 8m_vs_9m.

The experimental results clearly demonstrate the advantages of G2ACG's combination of group relationships and attention mechanisms. The algorithm not only incorporates GACG's approach to capturing group relationships, but also introduces dynamic attention mechanisms, enabling a more precise modeling of inter-agent relationships and consequently achieving superior performance compared to other algorithms.

5.2. Ablation Study

In this section, we conducted ablation experiments to further investigate the roles of various components in the introduced GATv2 network. In the 3s5z and 8m map environments, we examined the impact of different heads of attention on the learning performance of the GACG algorithm. Meanwhile, in the 3s5z and 8m_vs_9m map environments, we studied the performance variations of the G2ACG algorithm when modifying the number of GCN layers.

5.2.1. Heads for Dynamic Attention

To ensure that the output feature dimensions are divisible by the number of heads in our experiments, we uniformly set the message dimension coefficient to 0.5 (the message dimension is the observation dimension multiplied by this coefficient). Under this configuration, the experiment is conducted with h set to $\{1, 8\}$, allowing us to systematically investigate the impact of varying head numbers (h) on the algorithm's performance.

When $h = 1$, this indicates the absence of multiple attention mechanisms in the model, meaning that the model relies solely on a single perspective to capture relationships between nodes. Under this configuration, the model may fail to capture subtle interactions that can only be discovered by considering multiple aspects simultaneously. When $h = 8$, eight independent weights are employed, which independently compute attention scores and can capture node relationships from multiple different perspectives, thereby achieving a richer representation of the graph structure.

Figure 3 illustrates the impact of attention heads on G2ACG's performance. The results demonstrate that increasing the number of attention heads can improve the algorithm's learning effectiveness to some extent, which is particularly evident in the 3s5z map, while showing comparable performance between both agent groups in the 8m map.

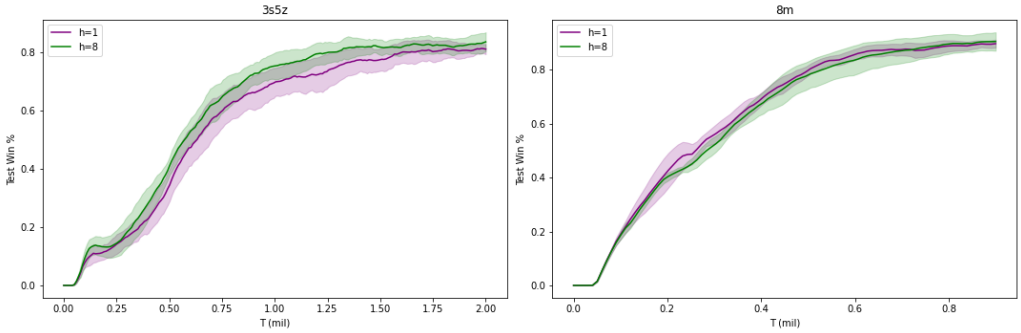

Figure 3. Experiment using different numbers of heads of attention (h).

5.2.2. Layers of GCNs

In this analysis, to thoroughly investigate the impact of GCN layer numbers (n) on algorithm performance, we explore different values for n, specifically $\{1, 2, 3, 4\}$. When $n = 2$, this represents our actually selected G2ACG algorithm configuration.

When $n = 1$, the model lacks deep hierarchical graph structure learning, capturing only direct neighborhood information for each node. Conversely, when $n \in \{2, 3, 4\}$, the presence of multiple GCN layers enables the model to capture deeper-level and more complex structural information.

The result is shown in Figure 4. According to the results, increasing the number of GCN layers within a certain range can improve the final win rate of the algorithm, while an excessively large number of layers leads to a decline in performance. In the scenarios of 3s5z and 8m_vs_9m, the learning performance reaches its optimal level with 2 layers. For an overly large number of GCN layers, on the one hand, as the number of layers

increases, the gradient during backpropagation becomes increasingly smaller, causing slow weight updates in the layers closer to the input, i.e., the vanishing gradient problem; on the other hand, information can be lost or overly smoothed during multi-layer propagation, making it difficult for the node feature representations to accurately reflect their intrinsic characteristics and the differences among neighboring nodes.

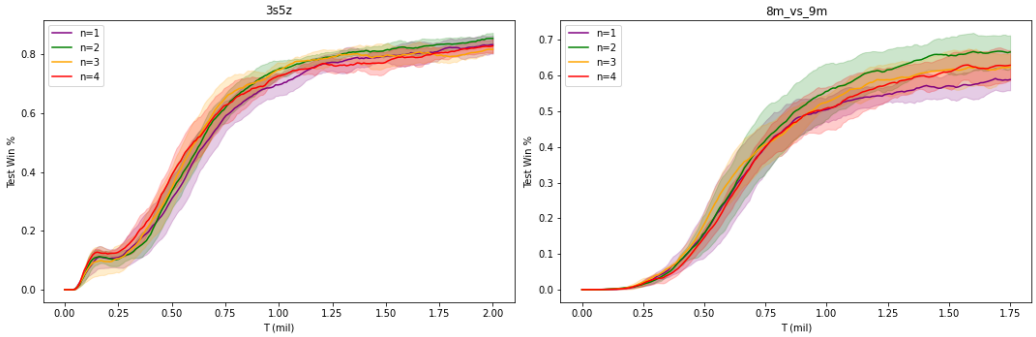

Figure 4. Experiment choosing different numbers of GCN layers (n).

6. Conclusions

In this paper, we propose G2ACG, which aims to solve the limitation that current multi-agent collaborative reinforcement learning algorithms cannot effectively combine group relationship utilization and node relationship capture, and aims to improve learning efficiency. Compared with previous algorithms, G2ACG not only integrates GACG's method of capturing group relations by observing behavior patterns on trajectories, but also constructs an edge Gaussian distribution with group relationships, which makes the information exchange efficiency of agents, through graph neural networks, higher in the decision-making process. At the same time, the latest dynamic attention mechanism is introduced to break the weight sharing mechanism in traditional GAT networks [11], so that the network can more flexibly model the varying preferences of nodes toward their neighbors and enhance the ability of the model to deal with complex environments. Through a series of comparative experiments, our approach is evaluated to be superior to the current baseline algorithm overall. In ablation experiments, we further explored the effects of numbers of heads and GCN layers on model performance. The results of this study can improve the accuracy of relationship capture between agents, and help to further improve the efficiency of multi-agent cooperation in more complex environments. Looking ahead, we will enhance G2ACG's scalability to hundreds of agents via hierarchical sparse graphs and distributed training, and extend it to real-world, communication-limited domains such as multi-robot coordination and autonomous driving fleets.

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

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: We are deeply grateful to our mentor for his meticulous guidance and selfless assistance throughout the research process, which enabled us to overcome one challenge after another. At the same time, we would also like to pay our highest respects to the pioneers in this field. It is their groundbreaking work that has laid a solid foundation for subsequent research and provided a wealth of ideas and methods, allowing us to stand on the shoulders of giants and further explore this field.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Zhang, K.; Yang, Z.; Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Cham, Switzerland, 2021; pp. 321–384.
2. Duan, W.; Lu, J.; Xuan, J. Group-Aware Coordination Graph for Multi-Agent Reinforcement Learning. In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, Jeju, Republic of Korea, 3–9 August 2024; Larson, K., Ed.; International Joint Conferences on Artificial Intelligence Organization: Jeju, Republic of Korea, 2024; Main Track; Volume 33, pp. 3926–3934. [CrossRef]
3. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning Based on Team Reward. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18, Stockholm, Sweden, 10–15 July 2018; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2018; pp. 2085–2087.
4. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 1–51.
5. Böhmer, W.; Kurin, V.; Whiteson, S. Deep coordination graphs. In Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, 12–18 July 2020; PMLR: New York, NY, USA, 2020; pp. 980–991.
6. Gupta, N.; Hare, J.Z.; Kannan, R.; Prasanna, V. Deep Meta Coordination Graphs for Multi-agent Reinforcement Learning. *arXiv* **2025**, arXiv:2502.04028. [CrossRef]
7. Brody, S.; Alon, U.; Yahav, E. How attentive are graph attention networks? *arXiv* **2021**, arXiv:2105.14491. [CrossRef]
8. Zhou, G.; Xu, Z.; Zhang, Z.; Fan, G. Mastering Complex Coordination Through Attention-Based Dynamic Graph. In Proceedings of the Neural Information Processing, Changsha, China, 20–23 November 2023; Luo, B., Cheng, L., Wu, Z.G., Li, H., Li, C., Eds.; Springer: Singapore, 2024; pp. 305–318.
9. 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 Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020; AAAI Press: Menlo Park, CA, USA, 2020; pp. 7211–7218. [CrossRef]
10. Samvelyan, M.; Rashid, T.; Schroeder de Witt, C.; Farquhar, G.; Nardelli, N.; Rudner, T.G.J.; Hung, C.M.; Torr, P.H.S.; Foerster, J.; Whiteson, S. The StarCraft Multi-Agent Challenge. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, 13–17 May 2019; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2019; pp. 2186–2188.
11. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018; Volume 1050, pp. 10–48550.
12. Kok, J.R.; Vlassis, N. Collaborative Multiagent Reinforcement Learning by Payoff Propagation. *J. Mach. Learn. Res.* **2006**, *7*, 1789–1828.
13. Nguyen, T.; Branke, J. Evolutionary Dynamic Optimization: A Survey of the State of the art. *Swarm Evol. Comput.* **2012**, *6*, 1–24. [CrossRef]
14. 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 Adaptive Agents and Multi-Agent Systems, Virtual, 3–7 May 2021; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2021.
15. Ruan, J.; Du, Y.; Xiong, X.; Xing, D.; Li, X.; Meng, L.; Zhang, H.; Wang, J.; Xu, B. GCS: Graph-Based Coordination Strategy for Multi-Agent Reinforcement Learning. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS '22, Virtual, 9–13 May 2022; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2022; pp. 1128–1136.
16. Duan, W.; Lu, J.; Xuan, J. Inferring Latent Temporal Sparse Coordination Graph for Multiagent Reinforcement Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *36*, 14358–14370. [CrossRef] [PubMed]

17. Finkelshtein, B.; Huang, X.; Bronstein, M.M.; Ceylan, I.I. Cooperative Graph Neural Networks. *arXiv* **2023**, arXiv:abs/2310.01267. [PubMed]
18. Goeckner, A.; Sui, Y.; Martinet, N.; Li, X.; Zhu, Q. Graph Neural Network-based Multi-agent Reinforcement Learning for Resilient Distributed Coordination of Multi-Robot Systems. In Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, United Arab Emirates, 14–18 October 2024; pp. 5732–5739. [CrossRef]
19. Oliehoek, F.A.; Amato, C. *A Concise Introduction to Decentralized POMDPs*; Springer: Cham, Switzerland, 2016; Volume 1.
20. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017; Conference Track Proceedings. Available online: https://OpenReview.net (accessed on 25 February 2025).