2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

GHGC: Goal-based Hierarchical Group Communication in Multi-Agent Reinforcement Learning

1st Hao Jiang
College of Computer
National University of Defense Technology
Changsha, China
jianghao18@nudt.edu.cn

2th Dianxi Shi
Tianjin Artificial Intelligence Innovation Center
National Innovation Institute of Defense Technology
Beijing, China
dxshi@nudt.edu.cn

3nd Chao Xue
Artificial Intelligence Research Center
National Innovation Institute of Defense Technology
Beijing, China
xuec11@tsinghua.org.cn

4rd Yajie Wang
College of Compute
National University of Defense Technology
Changsha, China
wangyajie18@nudt.edu.cn

5th Gongju Wang
Artificial Intelligence Research Center
National Innovation Institute of Defense Technology
Beijing, China
a1019569628@gmail.com

6th Yongjun Zhang
Artificial Intelligence Research Center
National Innovation Institute of Defense Technology
Beijing, China
yjzhang@nudt.edu.cn

***Abstract*—In large-scale multi-agent systems, the existence of a large number of agents with different target tasks and connected by complex game relationships causes great difficulty for policy learning. Therefore, simplifying the learning process is an important issue. In multi-agent systems, agents with the same target tasks or attributes often interact more with each other and exhibit behaviors more similar. That means there are stronger collaborations between these agents. Most existing multi-agent reinforcement learning (MARL) algorithms expect to learn the collaborative strategies of all agents directly in order to maximize the common rewards. This causes the difficulty of policy learning to increase exponentially as the number and types of agents increase. To address this problem, we propose a goal-based hierarchical group communication (GHGC) algorithm. This algorithm divides the agents into different groups, and maintains the group's cognitive consistency through knowledge sharing. Subsequently, we introduce a group communication and value decomposition method to ensure cooperation between the various groups. Experiments demonstrate that our model outperforms state-of-the-art MARL methods on the widely adopted StarCraft II benchmarks across different scenarios, and also possesses potential value for large-scale real-world applications.**

***Index Terms*—Multi-Agent Reinforcement Learning, Goal-based Hierarchical Group Communication, Group's Cognitive Consistency**

I. Introduction

Reinforcement learning (RL) combined with deep neural networks, has achieved human-level or higher performances in challenging games [1] [2]. The advances in RL and deep learning have led researchers to show great interest in MARL, hoping that it can resolve complex and large scale problems. MARL has shown great success in solving sequential decision-making problems with multiple agents.

Recently, more and more researchers are focusing on large-scale multi-agent reinforcement learning research [3] [4]. In large-scale multi-agent systems, a large number of agents with different goals and complex interactions between agents pose a significant challenge to the policy learning process. Therefore, simplifying the learning process is a crucial research area. Previous work focused on loosely coupled multi-agent systems, and adopt techniques such as game abstraction and knowledge transfer to help accelerate multi-agent reinforcement learning [5] [6] [7]. However, in a large-scale multi-agent environment, the agent is usually related to some other agents rather than independent, making the previously learned single-agent knowledge use limited. Recent work focuses on achieving game abstraction through predefined rules (e.g., the distance or communication between agents) [3] [8] [9]. However, in complex large-scale multi-agent systems, there are different types of interaction between the same target agents and different target agents. It isn't easy to describe and simplify the complex interaction relationships between agents in the system through distance or communication. In this work, we group agents according to their goals or attributes to reduce the complexity of interactions between agents and propose a hierarchical method to promote the learning of cooperative strategies.

The key to cooperative strategy learning in large-scale multi-agent systems is to learn the interaction between agents [10]. Recent works use "learning-for-consensus approach" [11] [12] and "learning-to-communicate approach" [13] [14] to learn the importance distribution of all agents i and other agents, in anticipation of learning directly the cooperation strategy of agent i with all other agents. However, in these two types of methods, the strategy learned by agent i is based on the information about all other agents with the same weight. In other words, these methods cannot reduce the complexity of the interaction between agents, and cannot ignore unrelated agents to simplify the learning process of cooperative strategies.

In this work, we propose a novel algorithm called GHGC (Goal-based Hierarchical Group Communication), which is trained end-to-end by backpropagation. It enables agents to learn different types of agent interactions in a partially observable distributed environment. It simplifies the learning process of large-scale multi-agent cooperation strategies and reduces the difficulty of strategy learning. The major contributions to this work are summarized as follows:

1) We use prior knowledge or data to gather all agents into different groups. Based on this, we propose a novel two-level environment abstraction mechanism for environmental cognitive consistency, which can ensure that the same group's agents have the same understanding of the environment. Environmental cognitive consistency is necessary to ensure good cooperation between agents.
2) To coordinate the cooperation of various groups, we introduce a group communication method and value decomposition method. The group communication can share part of the knowledge required for cooperation through information transmission when the environment of each group is inconsistent. Meanwhile, using the value decomposition method to establish the relationship between the joint action-value functions (Q_{tot}) of all agents and the joint action-value functions (Q_{te}) of the group agents to help ensure that the group goals can be achieved while collaborating to achieve collective goals.
3) Experiments are conducted in a challenging multi-agent collaborative environment with a high-dimensional state space: a variety of units in the StarCraft II battle game form a team to defeat their enemies. Test results show that GHGC outperforms state-of-the-art methods and shows its potential values for large-scale real-world applications.

II. Related Work

Many MARL methods focus on deriving decentralized policies (actors) for agents, each of which maps a local observation for an agent to an individual action for it. To make such individually chosen actions be coordinated to conduct collaborative tasks, these approaches first construct a centralized critic for either a global reward or individual reward and use the centralized critic to derive the decentralized actor. MADDPG [15] is an extension of the actor-critic model on the mixed cooperative-competitive environments. COMA [16] is proposed to solve multi-agent credit assignment for cooperative settings. MADDPG and COMA both use a centralized critic that takes all agents' observations and actions as input. However, MADDPG and COMA have to train an independent policy network for each agent, where each agent would learn a policy specializing specific tasks [17], and the policy network easily overfits to the number of agents. Therefore, MADDPG and COMA are infeasible in large-scale MARL.

Many other methods of decentralized policies on cooperative environments are using communication among agents. In this framework, each agent learns how to transmit messages to other agents and process the messages received from other agents to determine an individual action. During the centralized training, such message generating and processing procedures are learned to induce cooperation among agents. During the execution phase, agents exchange messages to determine their actions. CommNet [18] uses a vast single neural network to process all the messages transmitted by all agents globally, and the processed message is used to guide all agents to cooperate. BiCNet [19] is based on the actor-critic model for continuous action, using recurrent networks to connect each individual agent's policy and value networks. BiCNet can handle real-time strategy games such as StarCraft micromanagement tasks. IC3Net [20] has been proposed to actively select and mask messages from other agents during communication by applying gating function in the message aggregation step. However, all of these works directly construct a communication channel between all agents, which think that the interaction type of the agent i is the same as all other agents. This means that agent i has to deal with the cooperation information sent by all other agents. Therefore, this type of communication cannot achieve good collaboration in large-scale MARL.

III. Background

A. Partially Observable Markov Game

A Partially Observable Markov Game (POMG) is an extension of partially observable Markov decision process to a game with multiple agents. A POMG for N gents is defined as follows: $s \in \mathcal{S}$ denotes the global state of the game; $o_i \in \mathcal{O}_i$ denotes a local observation that agent i can acquire; $a_i \in \mathcal{A}_i$ is an action for agent i. The reward for agent i is computed as a function of state s and joint action $\mathbf{a}$ as $r_i : \mathcal{S} \times \mathcal{A}_1 \times \ldots \times \mathcal{A}_N \mapsto \mathbb{R}$. The state evolves to the next state according to the state transition mode $\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \ldots \times \mathcal{A}_N \mapsto \mathcal{S}$. The initial state is determined by the initial state distribution $\rho : \mathcal{S} \mapsto [0,1]$. The agent i aims to maximize its discounted return $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where $\gamma \mapsto [0,1]$ is a discount factor.

B. Deep Q-Learning

Deep Q-learning represents the action-value function with a deep neural network parameterised by θ. Deep Q-networks (DQNs) [1] use a replay memory to store the transition tuple (s, a, r, a'), where the state s' is observed after taking the action a in state s and receiving reward r. θ is learnt by sampling batches of b transitions from the replay memory and minimising the squared *TD error*:

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(Q(s, a; \theta) - y_i^{DQN} \right)^2 \right] \quad (1)$$

where $y^{DQN} = r + \gamma \max_{a'} Q(s', a', \theta')$. θ' are the parameters of a target network that are periodically copied from θ and kept constant for a number of iterations.

C. QMIX

QMIX learns a monotonic multi-agent Q-value approximation Q_{tot} [21]. QMIX factors the joint action-value Q_{tot} into a monotonic nonlinear combination of individual Q-value Q_i of each agent which learns via a mixing network. The mixing network with non-negative weights produced by a hypernetwork is responsible for combing the agent's utilities for the chosen actions into $Q_{tot}(\mathbf{s}, \mathbf{a})$. The decomposition allows for an efficient, tractable maximization as it can be performed linearly from decentralized policies as well as the easy decentralized employment. During learning, the QMIX agents use ϵ-greedy exploration over their individual utilities to ensure sufficient exploration.

IV. Method

In this section, we propose our novel algorithm GHGC, and make a comprehensive introduction. GHGC can achieve good cooperation in large-scale multi-agent systems that have a large number of agents with different targets and complex interactions between agents. The complete GHGC learning framework is shown in Fig.1. First, we use a priori knowledge to group agents in a multi-agent environment. Second, the two-level abstraction is proposed to obtain the agent's cognition of the environment in the group. It shares the low-level knowledge to achieve group cognition consistency, thereby achieving collaboration within the group. Third, we use a group communication and value decomposition method in order to achieve collaboration between the groups. Finally, we synthesize all of these above-mentioned components into a synergistic learning framework.

A. Agent Clustering

The first step is to cluster all of the N agents into distinct groups C^k using prior knowledge or data, such that $N = \sum_{k=1}^{K} |C^k|$. When a multi-agent system has only a single attribute or one type of target agents, all agents can be categorized into a single group. Moreover, if agents in the multi-agent system have multiple attributes or target tasks, we can cluster the agents into K groups according to the attribute or target task. K is the number of agent attributes or target tasks in the multi-agent system. In this work, we assume that the agents can be easily clustered into K groups using prior knowledge about the agents, which implies that GHGC utilizes enhanced relative inductive biases regarding the group relationships.

B. Group Cognitive Consistency

In large-scale multi-agent systems, agents with the same target or attribute will interact more and exhibit more similar behavior. This implies the necessity of stronger collaboration between agents with the same target or attribute. Agents that maintain consistent cognitions about their environments are vital if effective cooperation is to be achieved [22] [9] [23]. In this context, *cognition* refers to the agent's understanding of the local environment; this includes not only the observations of all agents in the same group, but also the high-level knowledge extracted from these observations (e.g., knowledge that has been learned through deep neural networks). Mao et al. [9] first proposed the method of neighborhood cognitive consistency (NCC). To implement NCC, these authors decompose the high-level knowledge into both an agent-specific cognitive representation and a neighborhood-specific cognitive representation. They then assume that each neighborhood contains a true hidden cognitive variable, after which they make all neighboring agents learn to align their neighborhood-specific cognitive representations to this true hidden cognitive variable using variational inference. MADDPG [15] uses a centralized critic capable of obtaining all agent actions and global observations in order to ensure that all agents receive the same environmental information. When all agents observe the same environmental information, it can be concluded that these agents have the same cognitive consistency. Adopting a different approach, in this work, we use the method of knowledge sharing to achieve group cognitive consistency.

There are N agents that can be divided K groups in a multi-agent environment. The agent i and agent j in same group $k = 1, \ldots, K$ are represented as C_i^k and C_j^k. Agent i has the local observation o_i. Under partially observable conditions, agent i computes the different node-embedding vectors h_i^k. The h_i^k that is encoded by MLP is a low-level cognition vector and contains only agent i's local observation, which can also be seen as an agent-specific cognitive representation. C_i^k and C_j^k have different agent-specific cognitive representations. In order to ensure that agents in the same group have consistent cognitions of the environment, we aggregate all h_i^k within the group k as input, then use LSTM to further extract the high-level cognition vectors H_i^k as follows:

$$H_i^k = LSTM(h_i^k, h_{-i}^k), where \quad h_i^k = f(o_i^k, a_i^k) \quad (2)$$

In Eq.2, $f(\cdot)$ is a fully connected layer for embedding. Moreover, h_{-i}^k is the low-level cognition vector of all other agents in group k except for agent C_i^k. We obtained the high-level cognitive vector H_i^k of each agent to the current environment through two-level encoding. When calculating the high-level cognitive vector H_i^k, we share the low-level cognition vectors h_i^k of the same group of agents. Therefore, H_i^k includes the low-level cognitions of all agents in group k. All the agents of the group k have consistent cognition of the environment.

C. Group Communication and Value Decomposition

Group cognitive consistency can achieve strong coordination within the group. However, proper cooperation between groups is also necessary. In order to achieve collaboration between the various groups, we allow communication between groups by the communication channel. The bi-directional LSTM unit acts as

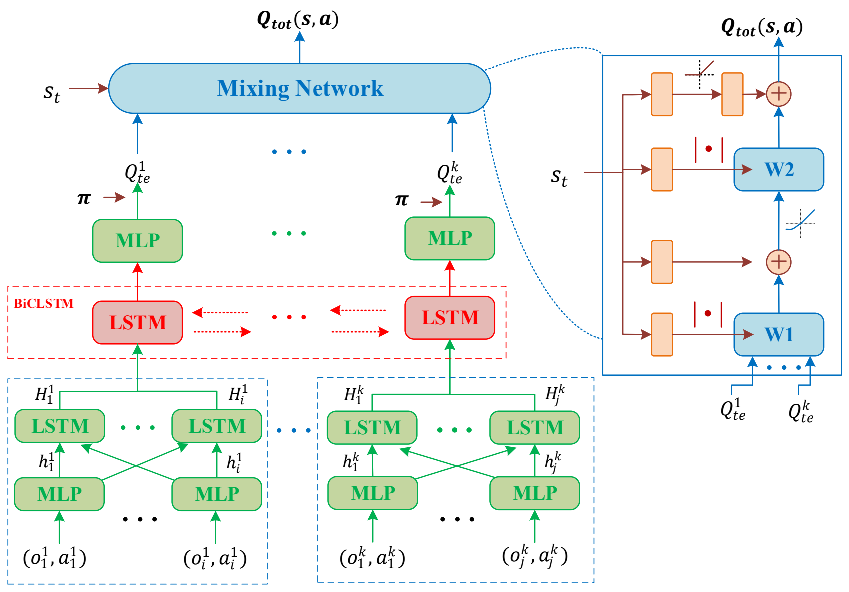


Fig. 1: Overview of GHGC

the communication channel, which takes as input internal states and returns thoughts that guide groups for coordinated strategies. Unlike CommNet and BiCNet that integrate shared information of agents by arithmetic mean and weighted mean, respectively, our LSTM unit can selectively output information that promotes cooperation and forget information that impedes cooperation through gates. We aggregate the high-level cognition vector $H_1^k, \ldots, H_i^k$ of group k as $V^k = \sum_{i=1}^{N_k} H_i^k$. The N_k is the number of agents in the group k. V^k is used as the input of the bi-directional LSTM to calculate the joint value $Q_{te}^k(\mathbf{o}^k, \mathbf{a}^k) = Q(V^k)$ of all agents in group k. Here $\mathbf{o}^k$ and $\mathbf{a}^k$ are the joint observation and the joint action of all agents in group k. In order to further strengthen the cooperation of various groups, we introduce the method of value decomposition. Q_{tot} is used to represent the joint value of all agents. Inspired by QMIX [21], we need to ensure that a global $\arg\max$ performed on Q_{tot} yields the same result as a set of each group's $\arg\max$ operations performed on each Q_{te}^k.

$$\arg\max_{\mathbf{a}} Q_{tot}(\mathbf{V}, \mathbf{a}) = \begin{pmatrix} \arg\max_{a^1} Q_{te}^1(V^1) \\ \vdots \\ \arg\max_{a^k} Q_{te}^k(V^k) \end{pmatrix} \tag{3}$$

where the $\mathbf{V} = \sum_{k=1}^{K} V^k$ and $\mathbf{a}$ are the joint observation and the joint action of all agents. The Q_{tot} and Q_{te}^k observe the monotonicity assumption [21] which can be generalised from Eq.3. The relationship between Q_{tot} and Q_{te}^k can be descraib as:

$$\frac{\alpha Q_{tot}}{\alpha Q_{te}^k} \geq 0 \tag{4}$$

To enforce Eq.4, GHGC represents Q_{tot} using a mixing network and a set of hyper networks [24]. As shown in Fig.1, the mixing network is a feed-forward neural network that takes the bi-directional LSTM network outputs as input and mixes them monotonically, producing the values of Q_{tot}. The weights (but not the biases) of the mixing network are restricted to being non-negative to enforce the monotonicity constraint Eq.4. This allows the mixing network to approximate any monotonic function arbitrarily closely [25]. We use ω denote the parameter of the mixing network, $Q_{tot} = \omega(Q_{te}^1, \cdots, Q_{te}^K)$.

GHGC is trained end-to-end to minimise the following loss:

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(Q_{tot}\left(\mathbf{V}, \mathbf{a}, s, \theta, \omega\right) - y^{tot} \right)^2 \right] \tag{5}$$

where b is the batch size of transitions sampled from the replay buffer, $y^{tot} = r + \gamma \max_{\mathbf{a}'} Q_{tot}(\mathbf{V}', \mathbf{a}', s', \theta', \omega')$ and θ' are the parameters of a target network as in DQN. ω' is the parameters of a target mixing network. Eq.5 is analogous to the standard DQN loss of Eq.1 Since Eq.4 holds, we can perform the maximisation of Q_{tot} in time linear in the number of agents (as opposed to scaling exponentially in the worst case).

V. Experiments

A. Settings

In this section, we evaluate GHGC on the StarCraft II decentralized micromanagement tasks, using the StarCraft Multi-Agent Challenge (SMAC) environment [26] as our testbed; this environment has become a commonly used benchmark for evaluating state-of-the-art MARL approaches such as COMA [16], QMIX [21] and BiCNet [19]. We train multiple agents to control allied units, while the enemy units are controlled by a built-in hand-crafted AI. At the beginning of each episode, the enemy units attack the allies. Proper micromanagement of units during battles is necessary to

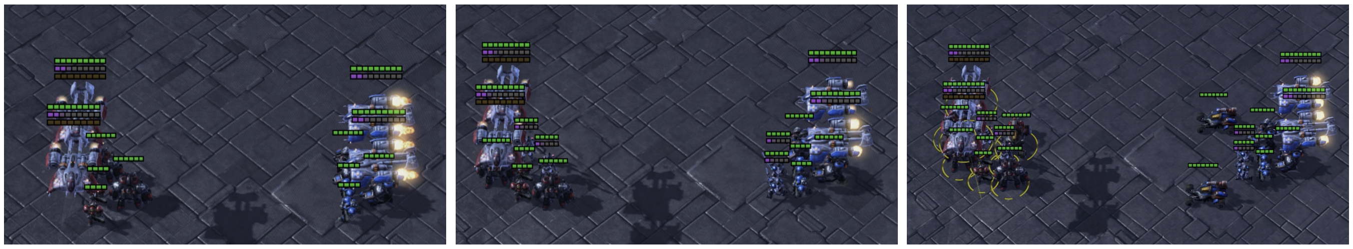


Fig. 2: The map scenarios of Map 1, Map2 and Map 3

TABLE I: Custom map

Name	Ally Units	Enemy Units	Type
Map1	2 Marines, 2 Medivacs, 2 Marauders	2 Marines, 2 Medivacs, 2 Marauders	Symmetric Heterogeneou
Map2	2 Marines, 2 Medivacs, 2 Marauders, 2Ghosts	2 Marines, 2 Medivacs, 2 Marauders, 2Ghosts	Symmetric Heterogeneou
Map3	2 Marines, 2 Medivacs, 2 Marauders, 2Ghosts, 2 Vikings	2 Marines, 2 Medivacs, 2 Marauders, 2Ghosts, 2 Vikings	Symmetric Heterogeneou

maximize the damage to enemy units while minimizing damage received, which requires a range of coordination skills such as focusing fire and avoiding overkill. Learning these different cooperative behaviors under partial observation conditions is challenging. Training and evaluation schedules, such as the testing episode number and training hyper-parameters, are kept the same as QMIX in SMAC.

All map scenarios have different unit numbers or unit types. We considered map scenarios contain 2 Stalkers & 3 Zealots (2S3Z), 3 Stalkers & 5 Zealots (3Z5Z), 5 Stalkers & 7 Zealots (5S7Z), Map1, Map2, and Map3. Here we briefly introduce the three map scenarios of Map1, Map2, and Map3 in Table 1 and show in Fig.2. In Map 3, it includes 10 units of Terran, which are 2 Marine, 2 Medivac, 2 Marauders, 2 Viking, and 2 Ghost. The attributes of each type of unit are described in Table 2. Each unit has its own positioning in the map scenario and plays different functions. The allied units can achieve victory only when each unit performs to the best of its ability and cooperates with the actions of other units.

The key winning strategy for Map 1 is that the Medivacs approach the enemies first, absorb fire and then retreat to heal the appropriate ally (i.e. the one with the least health). In Map3, moreover, we expect that the Vikings will be able to change their modes in a reasonable manner according to the situation on the battlefield. When the enemy's air power is strong, a Viking can enter a fighter mode that enables it to attack the enemy's aerial forces, conversely, when the enemy's ground strength is stronger than ours, the Vikings can switch to assault mode.

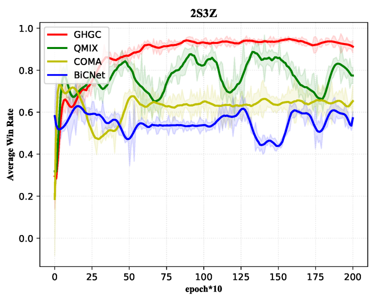
B. Expansion experiment of the number of agents

To evaluate the performance and verify that our algorithm can easily expand the number of agents, we conduct experiments on three map scenarios: 2S3Z, 3S5Z, and 5S7Z. The results are presented in Fig.3; here, each experiment is repeated five times to enable plotting of the winning rate curve with standard deviation. These results demonstrate that GHGC outperforms QMIX and other popular methods. GHGC can master these tasks and achieve a winning rate of around 90% in the three map scenarios. In 2S3Z (see Fig.3a), the win rate of the QMIX, COMA, and BiCNet algorithms can reach about 80%, 63%, and 58% respectively, indicating that these algorithms can learn a good policy for the agents on scenario 2S3Z. On 3S5Z (see Fig.3b), moreover, the win rate of the QMIX, COMA, and BiCNet algorithms are 47%, 45%, and 44%, respectively; this indicates that these algorithms may fail to learn a good policy for the agents on scenario 5S7Z (see Fig.3c).

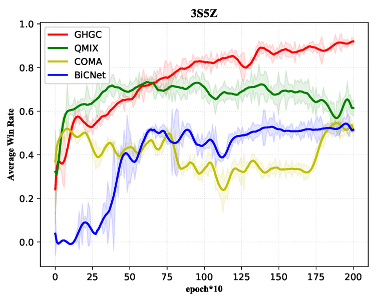
The difficulty experienced by agents in learning strategy increases with the number of agents. As shown in Fig.4a, as the number of units increases, the baseline algorithms of QMIX, COMA, and BiCNet achieve worse performance, while their winning rate also decreases. However, the GHGC algorithm can still maintain a high win rate and learn the correct strategies even if the number of agents increases. In order to further explore the trend of algorithm performance decreasing along with the number of agents, we normalize the reward to obtain the score of QMIX, COMA, and BiCNet compared with GHGC in for same scene. Experimental results (see Fig.4b) show that the scores of QMIX, COMA, and BiCNet are lower than that of GHGC with the same increase in agent quantity. This demonstrates that our GHGC algorithm has the potential to be applied to a large number of agents.

C. Agents type expansion experiments

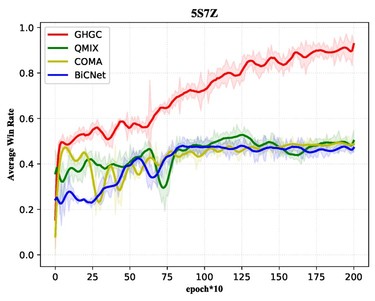
To evaluate performance and verify that our algorithm has good type scalability, we conduct further experiments on the three scenarios shown in Fig.2. In these three scenarios, as the unit type complexity gradually increases, the agent strategy learning becomes more and more difficult, while the interactions between various types of agents also become increasingly different and complex. Taking Medivac and Marauder as an example, the type of interaction that takes place between the two Medivacs is different from that between the Medivac and the Marauder from a unit function perspective. The interaction of two Medivacs takes the form of cooperating on a treatment



(a) 2S3Z

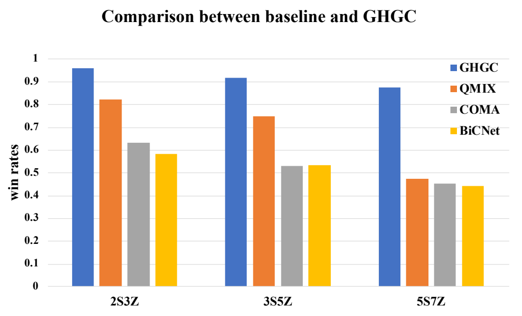


(b) 3S5Z

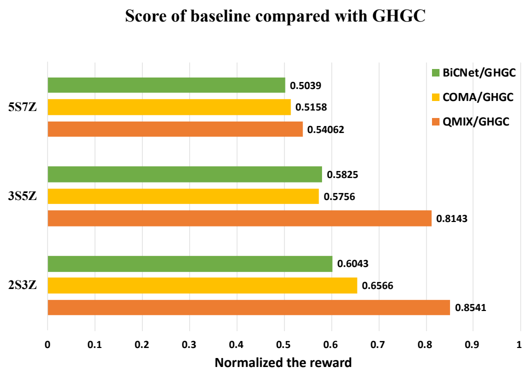


(c) 5S7Z

Fig. 3: The average win rates for COMA, BiCNet, QMIX and GHGC on three combat maps with different numbers of agents.

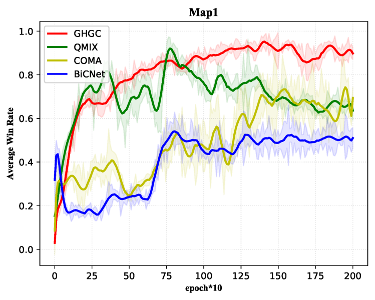


(a) The average win rate

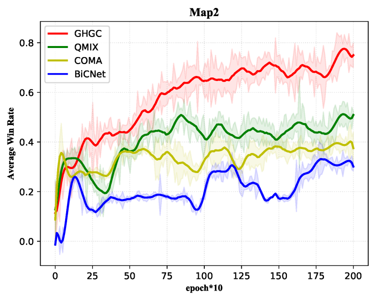


(b) Normalized the reward

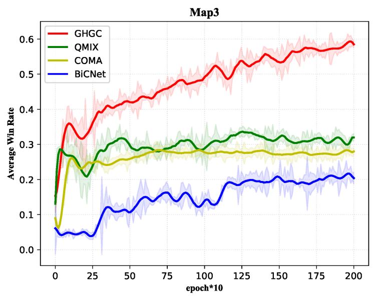
Fig. 4: Comparison between baseline algorithms and the GHGC algorithm on three combat maps (2S3Z, 3S5Z, 5S7Z). (b): each bar cluster shows the 0-1 normalized score for rewards of learned policies with different numbers of agents. As the number of cooperating agents increases, our GHGC algorithm achieves increasingly better performance compared with baseline algorithms.



(a) Map 1



(b) Map 2

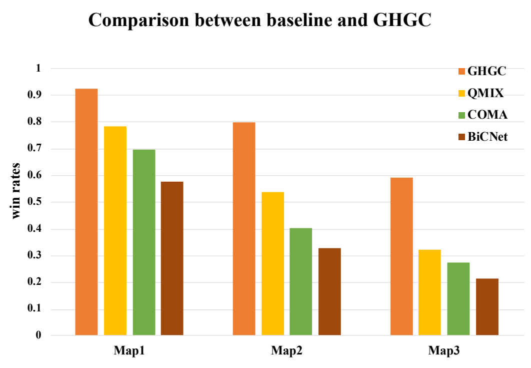


(c) Map 3

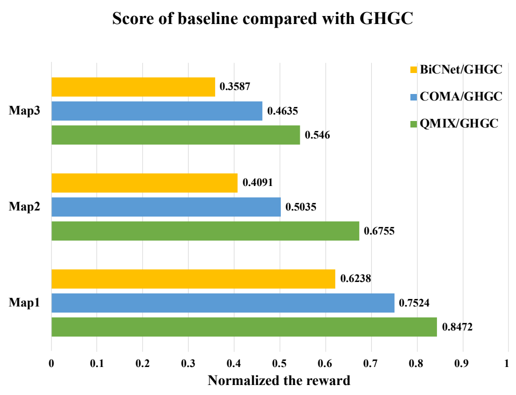
Fig. 5: The average win rates for COMA, BiCNet, QMIX and GHGC on three combat maps (Map1, Map2, Map3) with different types of agents.

TABLE II: The attributes of each type of units

Name	Function	Type
Medivac	**Healing ability**, it can provide treatment to the injured unit; **Transportation ability**, it can withdraw the unit from the battlefield.	Terran
Marauder	**Slowing ability**, it can reduce the speed of enemy units and pursue the enemy; **High defense**, it can withstand more attacks from the enemy.	Terran
Marines	Its low cost is suitable for mass production.	Terran
Ghosts	**Stealth ability**, it can disappear in the enemy's field of vision; **Sniper ability**, it can range the enemy units at long range.	Terran
Viking	**Assault Mode**, it can only attack ground units; **Fighter Mode**, it an only attack air units.	Terran



(a) The average win rate



(b) Normalized the reward

Fig. 6: Comparison between baseline algorithms and the GHGC algorithm on three combat maps (Map1, Map2, Map3). (b) each bar cluster shows the 0-1 normalized score for rewards of learned policies with different types of agents. As the number of agent types increases, our GHGC algorithm achieves increasingly better performance compared with baseline algorithms.

task to achieve the maximum amount of treatment on the battlefield. By contrast, the interaction between Medivacs and Marauders takes the form of Marauders protecting the Medivacs from being killed, while Medivacs protect the Marauders from damage and provide them with treatment.

The results are presented in Fig.5, which is repeated 5 times to plot the winning rate curve with standard deviation. These results demonstrate that GHGC outperforms QMIX and other popular methods. Even if there are many units with complex types and different capabilities on the battlefield, our GHGC algorithm is still able to achieve good cooperation between all units and win the battle. As shown in Fig.6a, as the scenario complexity increases, the performance of the GHGC, QMIX, COMA, and BiCNet algorithms deteriorates. As shown in Fig.6b, the performance of the QMIX, COMA, and BiCNet algorithms also deteriorates more rapidly than that of the GHGC algorithm. In the Map 1 scenario (see Fig.5a), the GHGC algorithm can achieve a win rate of about 90%. In the hard Map 3 scenario (see Fig.5c), moreover, it can achieve a win rate of about 60%.

Furthermore, the win rates of the QMIX, COMA, and BiCNet algorithms are lower than that of GHGC in all three scenarios, especially in the Map 3 scenario; in short, for all these scenarios, GHGC consistently achieves the best performance. This demonstrates that considering the different interactions between various types of agents can ultimately induce better-trained policies.

VI. Conclusion

In this work, we focus on simplifying policy learning in large-scale multi-agent systems. GHGC can reduce the complexity of interaction between various agents and facilitate the learning of between-agent relationships. Empirically, we demonstrated that the learned model outperformed other MARL models in a variety of multi-agent environments. Moreover, the GHGC algorithm has been proven to expand the number and types of agents easily, and may therefore be valuable for large-scale real-world applications. In future work, we plan to carry on experiments of letting the machine compete with human players at different levels.

VII. Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant No.2017YFB1001901, in part by the Key Program of Tianjin Science and Technology Development Plan under Grant No.18ZXZNGX00120 and in part by the China Postdoctoral Science Foundation under Grant No.2018M643900.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," *arXiv preprint arXiv:1802.05438*, 2018.

[4] Y. Chen, M. Zhou, Y. Wen, Y. Yang, Y. Su, W. Zhang, D. Zhang, J. Wang, and H. Liu, "Factorized q-learning for large-scale multi-agent systems," *arXiv preprint arXiv:1809.03738*, 2018.

[5] F. S. Melo and M. Veloso, "Decentralized mdps with sparse interactions," *Artificial Intelligence*, vol. 175, no. 11, pp. 1757–1789, 2011.

[6] C. Yu, M. Zhang, F. Ren, and G. Tan, "Multiagent learning of coordination in loosely coupled multiagent systems," *IEEE transactions on cybernetics*, vol. 45, no. 12, pp. 2853–2867, 2015.

[7] Y. Liu, Y. Hu, Y. Gao, Y. Chen, and C. Fan, "Value function transfer for deep multi-agent reinforcement learning based on n-step returns," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019, pp. 457–463.

[8] J. Jiang, C. Dun, and Z. Lu, "Graph convolutional reinforcement learning for multi-agent cooperation," *arXiv preprint arXiv:1810.09202*, vol. 2, no. 3, 2018.

[9] H. Mao, W. Liu, J. Hao, J. Luo, D. Li, Z. Zhang, J. Wang, and Z. Xiao, "Neighborhood cognition consistent multi-agent reinforcement learning," *arXiv preprint arXiv:1912.01160*, 2019.

[10] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," *arXiv preprint arXiv:1911.10715*, 2019.

[11] Z. Zhang, J. Yang, and H. Zha, "Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization," *arXiv preprint arXiv:1909.10651*, 2019.

[12] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," *arXiv preprint arXiv:1810.02912*, 2018.

[13] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," in *Advances in neural information processing systems*, 2018, pp. 7254–7264.

[14] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," *arXiv preprint arXiv:1810.11187*, 2018.

[15] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[16] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[17] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1995–2003.

[18] S. Sukhbaatar, R. Fergus *et al.*, "Learning multiagent communication with backpropagation," in *Advances in neural information processing systems*, 2016, pp. 2244–2252.

[19] R. Henriques and S. C. Madeira, "Bicnet: Flexible module discovery in large-scale biological networks using biclustering," *Algorithms for Molecular Biology*, vol. 11, no. 1, p. 14, 2016.

[20] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," *arXiv preprint arXiv:1812.09755*, 2018.

[21] S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi- agent reinforcement learning," 2018.

[22] A. OroojlooyJadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *arXiv preprint arXiv:1908.03963*, 2019.

[23] A. Bear, A. Kagan, and D. G. Rand, "Co-evolution of cooperation and cognition: the impact of imperfect deliberation and context-sensitive intuition," *Proceedings of the Royal Society B: Biological Sciences*, vol. 284, no. 1851, p. 20162326, 2017.

[24] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[25] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating functional knowledge in neural networks," *Journal of Machine Learning Research*, vol. 10, no. Jun, pp. 1239–1262, 2009.

[26] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser *et al.*, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.