Temporal Dynamic Weighted Graph Convolution for Multi-agent Reinforcement Learning

Yuntao Liu (liuyuntao.me@gmail.com), Yong Dou (yongdou@nudt.edu.cn)
National University of Defense Technology
Hunan Changsha, China

Yuan Li, Xinhai Xu* ({yuan.li,xuxinhai}@nudt.edu.cn), Donghong Liu* (liudonghong@sina.com)
Academy of Military Sciences
Beijing, China

Abstract

In many real-world settings, it is crucially vital for agents to learn to communicate and cooperate. Different cooperation models have been proposed to represent cooperative relations among agents. However, the intensity of the cooperative relation has not received much attention. In particular, how it varied with spatial-temporal information has not been studied deeply. In this paper, we propose a temporal dynamic weighted graph convolution based multi-agent reinforcement learning framework (TWG-Q). We design a weighted graph convolutional network to capture cooperative information among agents. On top of that, a temporal weight learning mechanism is introduced to characterize intensities of cooperations. We design a novel temporal convolutional network in the temporal dimension to extract effective features for the multi-agent reinforcement learning. Extensive experiments show that our method significantly improves the performance of multi-agent reinforcement learning on the public benchmark of micromanagement tasks in StarCraft II.

Keywords: multi-agent reinforcement learning; graph convolutional network; temporal convolutional network

Introduction

Cooperation among agents performs an important role in multi-agent reinforcement learning (MARL)(Foerster, Assael, De Freitas, & Whiteson, 2016; Kim et al., 2019; Tung, Pujol, Kobus, & Gunduz, 2021; W. Z. Wang, Shih, Xie, & Sadigh, 2022). Recently, value decomposition MARL methods, i.e., Value Decomposition Network(VDN) (Sunehag et al., 2018) and QMIX (Rashid et al., 2018) are proposed for learning cooperation in MARL. Value decomposition methods make agents cooperate with each other to accumulate higher reward by maximizing the global Q-value which is a combination of individual Q-values that condition on partial observations and individual actions. However, there is less cooperation modeling in value decomposition MARL methods (OroojlooyJadid & Hajinezhad, 2019). The value decomposition mechanism can only lead agents to act in an individual way instead of a cooperative way (Jiang, Dun, Huang, & Lu, 2020; Nguyen, Nguyen, & Nahavandi, 2020).

To address this problem, communication-based MARL methods have been proposed to promote cooperations among agents. Some early works (Sukhbaatar, Szlam, & Fergus, 2016; Peng et al., 2017; Kong, Xin, Liu, & Wang, 2017) consider using the communication channel to share observations among all agents. In such cases, an agent could be flooded by information as the number of agents grows. (Zambaldi et al., 2018; Tacchetti et al., 2018; Mao et al., 2020) make some restrictions that each agent only communicate with its neighbors. However, it is normally hard to choose proper neighborhoods of agents in many complex applications which contain different roles. Moreover, those methods rely on message transmitting to communicate, which lacks explicit models for cooperative relations among agents.

In recent years, graph neural networks have been regarded as a useful tool to model cooperations for multi-agent reinforcement learning. DGN (Jiang et al., 2020) introduces the graph convolutional operation to reveal cooperative features among agents. (Böhmer, Kurin, & Whiteson, 2020) introduces the framework of multi-agent reinforcement learning with a coordination graph (CG), which could factorize the joint value function of all agents into payoffs between pairs of agents. (Li, Gupta, Morales, Allen, & Kochenderfer, 2021) proposes the deep implicit CG architecture, which generates a dynamic CG based on a self-attention network. Similarly, (Iqbal & Sha, 2019) introduces the self-attention mechanism for the cooperative weight learning, which takes shared observations and actions of all agents as inputs. (Liu et al., 2020) designs a two-stage weight learning mechanism on the graph to indicate the importance of the interaction between two agents. The weight learning mechanism relies on features extracted by the bi-direction LSTM module. However, these methods only consider features at the current time step and ignore features in the temporal dimension. Temporal features are important for the cooperative weight learning because numerous cooperative behaviors may last for a period. This is the main starting point of our study.

In this paper, we propose a temporal cooperative weighted graph convolution MARL method, called TWG-Q, to fit with the drastic changing cooperation among agents in MARL systems. Different with (Sukhbaatar et al., 2016), we design a weight learning mechanism for valuable cooperative information revealing. Our work is more similar to (Liu et al., 2020), but we improve the weight learning mechanism by introducing temporal cooperative features. Firstly, we construct the multi-agent environment as a weighted graph to model relations among agents. Nodes represent agents in the environment and edges correspond to relationships among agents. We introduce the weighted graph convolutional

This work was supported in part by the National Natural Science Foundation of China under Grant 61902425

* Corresponding author

In J. Culbertson, A. Perfors, H. Rabagliati & V. Ramenzoni (Eds.), *Proceedings of the 44th Annual Conference of the Cognitive Science Society.*

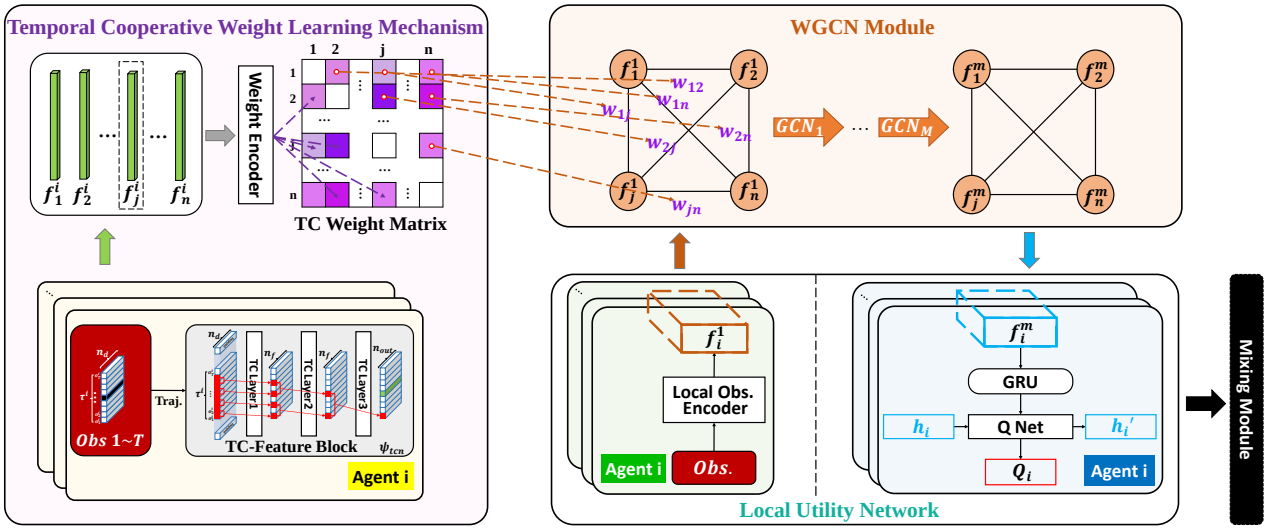


Figure 1: The architecture of TWG-Q.

network (GCN) (Kipf & Welling, 2016) to implement the feature interaction for cooperations. Then we design a temporal cooperative weight learning mechanism to compute weights on all edges in the graph. In the temporal dimension, we introduce a Temporal Convolutional Network (TCN) (Lea, Flynn, Vidal, Reiter, & Hager, 2017; Bai, Kolter, & Koltun, 2018; Zatsarynna, Abu Farha, & Gall, 2021) to extract features over a certain time range. The receptive field technique in TCN improves the efficiency of temporal feature extraction greatly. Finally, we compute accurate weights based on the temporal cooperative weight learning mechanism.

With TWG-Q, we could generate dynamical weights on the graph based on the varying temporal information. The generated weights could accurately character cooperative relations among agents, which greatly improves the performance of MARL methods. We make extensive experiments on some challenging micromanagement tasks in StarCraft II. The results show that TWG-Q performs much better than other popular cooperation MARL methods, which makes a significate step forward in graph convolution based MARL methods.

Background

Markov Games

Multi-agent reinforcement learning (MARL) could be modelled by a multi-agent extension of Markov Decision Processes (Yang et al., 2020). It is represented by $< N, S, \{A_i\}_{i=1}^N, \{R_i\}_{i=1}^N, T >$, where N is the number of agents. S is the state space, A_i is the action of agent $i (i = 1, \cdots, N)$, $R_i : S \times A_1 \times \cdots \times A_N \to R$ is the reward function of agent i and $T = S \times A_1 \times \cdots \times A_N \to [0,1]$ is the state transition function. We are specially concerned with a partially observable Markov game. Each agent i in the partially observable Markov game gets a local observation o_i and learns a policy $\pi_i : o_i \to P(A_i)$, where $P(A_i)$ is a distribution of the action space. The policy maps the local observation of each agent to a distribution of its action set. The goal of each agent i is to maximize its reward $R_i = \sum_{t=0}^{T} \gamma^T r_i^T$, where $\gamma \in [0,1]$ is the discounted factor. In our work, we consider a fully cooperative setting where each agent i receives a partial observation o_i. The action of each agent a_i is computed with its policy π_i based on o_i at time step t. A reward r is obtained from the environment after the joint action of all agents is executed. The reward is used to compute the total loss to update the neural network.

Graph Convolutional Network

Graph Convolutional Networks (GCNs) introduce convolutions on graphs. Each GCN layer applies message passing to compute a node representation, where each node aggregates feature vectors of neighboring nodes. Formally, a GCN model consists of K-layers and the k-th layer is implemented as Equation 1

$$X^{(k)} = ReLU(AX^{(k-1)}W^{(k)}) \tag{1}$$

where A is the adjacency matrix of the graph. $X^{(k)} \in \mathbb{R}^{N \times d_k}$ is the hidden feature matrix at layer k. $X^{(0)} \in \mathbb{R}^{N \times d}$ is the input observation vector. The parameter matrix of layer k is represented as $W^{(k)} \in \mathbb{R}^{d_k \times d_k}$. In the GCN model, features are often fed into an activation function before they are forwarded to the next layer.

In this paper, we model the communication model by the GCN in the cooperative MARL approach. The graph is undirected which is represented by $G = (V, E)$. V is the vertex set and each vertex $v_i \in V$ represents an agent entry in the environment. Each edge e_{ij} in the edge set E connects two agents $\{a_i, a_j\}$, which reflects the cooperation relation between agent a_i and agent a_j. For the GCN model, each agent can communicate with each other by passing their

local observations, which could improve the performance of MARL methods.

Methods

In this section, we propose a novel cooperative MARL framework based on the temporal weighted graph convolution network. In the following, we first describe the overall architecture of our framework. Then we detail two important components of our framework, i.e., the temporal cooperative weight learning mechanism and the weighted graph convolution model.

Overall Architecture

Our framework includes the local utility network, the temporal cooperative weight learning module and the communication module. The local utility network consists of a local observation encoder and an action generator. The local observation encoder contains a Multilayer Perceptron (MLP) φ_{obs}. It encodes the local observation o_i^t to agent's local feature $f_i^t = \varphi_{obs}(o_i^t)$. The communication module involves a weighted graph convolutional network (WGCN) ϕ_{gcn} and a temporal cooperative weight learning module φ_w. The temporal cooperative weight learning module φ_w generates weight w_i^t for WGCN based on temporal cooperative features extracted from the Temporal Convolutional Network (TCN) model ψ_{tcn}. The WGCN model ϕ_{gcn} takes graph adjacent matrix A, the temporal cooperative weight w_i^t and the local feature f_i^t as inputs and computes the output features g_i^t with cooperative information. The action generator includes a Gated Recurrent Unit (GRU) ψ_{gru} and a Q-value network φ_q. It takes the local feature f_i^t, the cooperative feature g_i^t and the historical information h_i^{t-1} as inputs. Q-values $Q_i^t(f_i^t, g_i^t, h_i^{t-1}, a_i)$ for all agents are computed based on local observations and features with cooperative information captured by the WGCN model. The introduction of the communication model based on the temporal cooperative weight learning mechanism is beneficial for the action selection. The reason is that each agent can communicate with each other to exchange temporal cooperative information. The agent network architecture of all other agents is similar to the above-mentioned architecture. During training, the action is selected with the ε-greedy policy (Wunder, Littman, & Babes, 2010). To generate the joint Q-value function Q_{total} for computation of loss objective, we introduce the mixing network to mix up all individual Q-values $Q_i^t(f_i^t, g_i^t, h_i^{t-1}, a_i)$. The architecture of the mixing network is similar to that in QMIX (Rashid et al., 2018). It introduces the value decomposition mechanism into our communication method, which allows each agent to get the suitable reward based on the joint reward to achieve better performance.

Temporal Cooperative Weight Learning Mechanism

To generate an accurate representation of cooperative relations among agents, we introduce a weight learning mechanism into our WGCN module to generate weights over time to deal with dynamic cooperative relations.

The designed TC weight learning mechanism models the temporal cooperation based on historical trajectories. The TC weight learning mechanism module takes the observation trajectory τ_i for each agent i as input and outputs the weight vector w_i. Each element w_{ij}^t represents the weight between agent i and j at time step t. The weight of the cooperation between any two agents is related to the temporal cooperative features. The *TCN* model ψ_{tcn} is applied on the whole trajectory τ_i to generate temporal cooperative features $\mathbf{f_{tc\_i}}$. Then we forward the temporal cooperative features $f_{tc\_i}^t$ to the weight network φ_w and compute the TC weight w_i^t for agent i at time step t.

Therefore, we introduce a weight learning mechanism into the communication model to generate weights over time to deal with dynamic cooperation relations among agents in the environment. To generate an accurate representation of cooperative relations, we consider dynamics cooperative features.

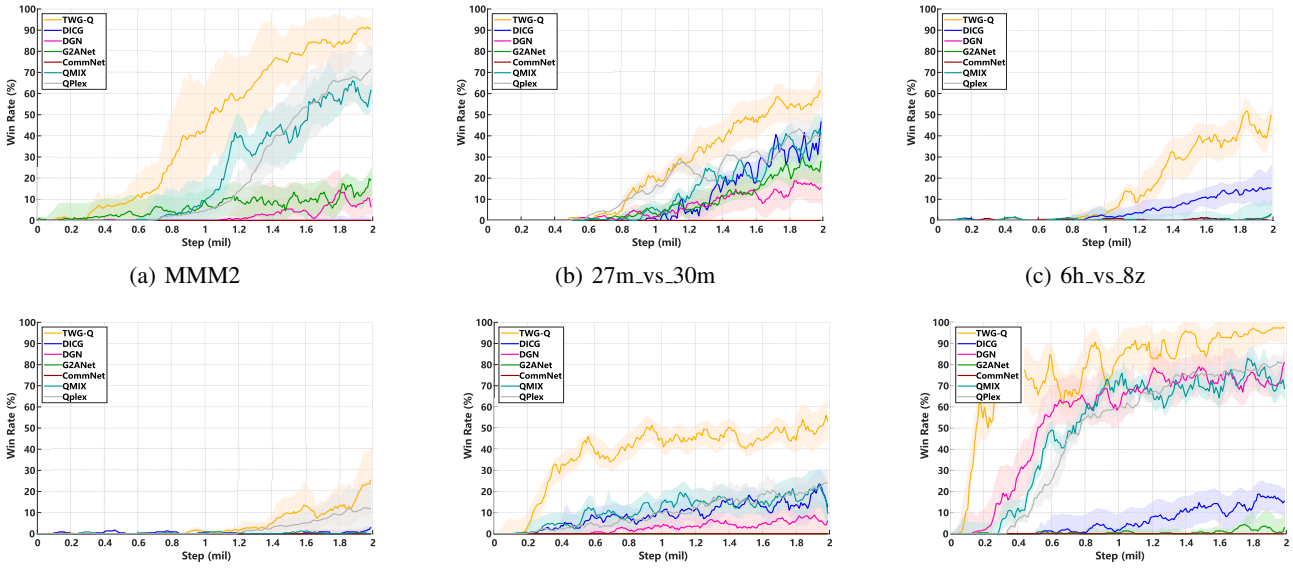
Now we detail the vital component temporal cooperative features extractor, namely the TCN model. In the temporal dimension, the long-term behaviors of agents are critical for cooperative multi-agent systems, which is often hidden in trajectories of agents. RNN and its variants (Mikolov, Karafiát, Burget, Cernockỳ, & Khudanpur, 2010) encode the historical trajectory one by one in the temporal dimension, and generate the hidden state to integrate history information. However, the hidden state at time step t includes all historical information before t, which makes it hard to extract valuable features. Moreover, many cooperative behaviors only occur in a period of time that is much smaller than the duration of an entire episode. Therefore, for the temporal module, it is crucial to concentrate on local temporal features over a certain time range, which is hard to be disposed with RNN models. Inspired by the receptive field technique used in the CNN, we introduce the Temporal Convolutional Network (TCN) as the temporal module. With convolutional operations, the TCN model has flexible receptive field on temporal features, which makes it suitable to encode the cooperative information contained in local temporal features.

A TCN model consists of N temporal convolutional (TC) layers, which can be described as $\psi_{tcn} = \{l_1, l_2, \cdots, l_N\}$. Each TC layer consists of a dilated $1D$ convolutional filter $W_{Ft} : \{0, \cdots, k-1\} \to \mathbb{R}$, the ReLU activation $ReLU(\cdot)$ and a residual connection $W_{Fr} : \{0, \cdots, k-1\} \to \mathbb{R}$. Formally, the dilated convolutional operation F on $\mathbf{x}_i$, the i-th element of the sequence feature $\mathbf{x} \in \mathbb{R}^n$, can be described as

$$F(i) = (\mathbf{x} * W_{Ft})(i) = \sum_{j=0}^{k-1} W_{Ft}(j) \cdot \mathbf{x}_{i-d \cdot j} \tag{2}$$

where d is the dilation factor, k is the filter size and $i - d \cdot j$ represents the past features.

The input for each TC layer l_k is the output from the former TC layer l_{k-1}. Particularly, the first TC layer takes agents' historical trajectories as input features. The activations in the k-th TC layer are given by $H_{TCN_k} \in \mathbb{R}^{n_f \times T}$, which can



(a) MMM2 (b) 27m_vs_30m (c) 6h_vs_8z

(d) 3s5z_vs_3s6z (e) 1m2h_vs_8m (f) nuke_vs_cannon

Figure 2: Comparison of TWG-Q with popular MARL algorithms on six different maps.

be described as follows:

$$\widehat{H}_{TCN\_k} = ReLU(W_{Ft} * H_{TCN\_k-1} + b_t) \tag{3}$$

$$H_{TCN\_k} = H_{TCN\_k-1} + W_{Fr} * \widehat{H}_{TCN\_k} + b_r \tag{4}$$

where $*$ denotes the convolutional operator. $W_{Ft} \in \mathbb{R}^{3\times n_f \times n_f}$ are the weights of the dilated convolutional filters with kernel 3 and n_f represents the number of those filters. $W_{Fr} \in \mathbb{R}^{1\times n_f \times n_f}$ are the weights of 1D convolution in the residual connections. $b_t, b_r \in \mathbb{R}^{n_f}$ are bias vectors. The receptive field can be increased by dilated convolution and increasing TC layers. Formally, the receptive field at the k-th TC layer is related to the layer number, which can be determined as Equation 5:

$$rf(k) = 3^{k+1} - 1 \tag{5}$$

Weighted Graph Convolutional Network

In this section we design a weighted graph convolutional network (WGCN) to model cooperations among agents.

In our WGCN model, the TC weights are organized as an adjacency matrix. Suppose A is the original adjacency matrix of GCN. Notice that we construct a complete agent graph, so all elements in A equal to 1. Each row in the adjacency matrix B of the WGCN model could be computed by:

$$B_i^t = w_i^t \circ A_i^t \tag{6}$$

where w_i^t is the TC weight computed in previous section and $\circ$ represents the element-wise multiplication between two vectors. Meanwhile, we add the identity matrix I to keep the feature of the agent itself, inspired by the idea of self-looping (Kipf & Welling, 2016).

We now detail the architecture of our WGCN module ϕ_{gcn}^j. Here $j \in \{1, \cdots, m\}$, where m is the number of layers in WGCN. Each ϕ_{gcn}^j takes the output feature H_{WGCN}^{j-1} of the former WGCN layer as input and outputs a vector H_{WGCN}^j. The main body of the model then computes:

$$H_{WGCN}^j = \sigma(C_i(B+I)H_{WGCN}^{j-1}W) \tag{7}$$

Particularly, the input H_{WGCN}^0 of the first WGCN layer is the combination of local features of all agents:

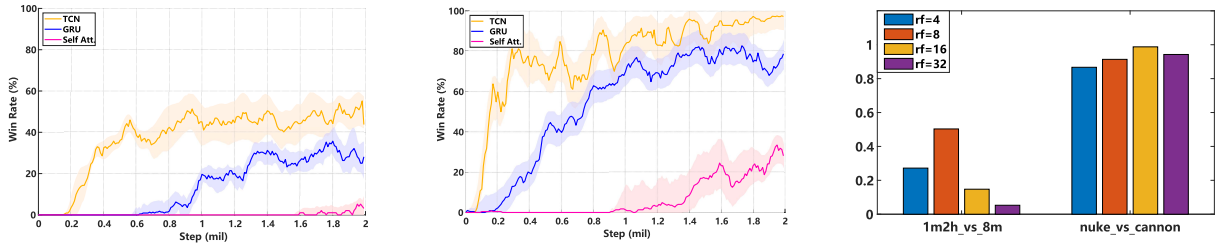
$$H_{WGCN}^0 = \{f_0; f_1; \cdots; f_N\} \tag{8}$$

where $C_i(\cdot)$ is the clip function for the adjacency matrix, H_o is the output of the local observation encoder, W is the parameters of WGCN and σ is the non-linear $ReLU$ function.

The WGCN model combines the cooperative information with the local observation of agents and outputs high quality features H_{WGCN} for the follow-up training of our TWG-Q model.

Experiments

In this section, we evaluate the performance of the proposed Temporal Weighted Communication based MARL framework (TWG-Q) in decentralized micromanagement tasks in the StarCraft Multi-Agent Challenge (SMAC) environment (Samvelyan et al., 2019), a public benchmark for testing state-of-the-art MARL approaches. Maps in SMAC are divided into three levels, i.e., easy, hard and super hard. In our experiments, we choose four super hard scenarios, i.e., (*MMM2*, *27m_vs_30m*, *6h_vs_8z* and *3s5z_vs_3s6z*). To examine the performance of TWG-Q in the temporal dimension, we design two special scenarios, *1m2h_vs_8m* and *nuke_vs_cannon*, with significant temporal characters. All experiments are conducted with 5 different random



(a) 1m2h_vs_8m (b) nuke_vs_cannon (c) different *rf*s

Figure 3: Comparison of different temporal modules and receptive fields.

seeds. The number of training, evaluation episodes and other hyper-parameters are kept the same as QMIX in SMAC environment. Our TCN model consists of 3 TC layers and WGCN model includes 2 WGCN layer.

Performance Evaluation

In this subsection, we compare TWG-Q with some state-of-the-art communication based methods and the representative value decomposition method. CommmNet (Sukhbaatar et al., 2016) is a fundamental one in which MLP is used to deal with features of all agents. DICG (Li et al., 2021) and DGN (Jiang et al., 2020) use the coordination graph network and the graph convolution network to model relations among agents respectively. In G2ANet (Liu et al., 2020), weights are taken into consideration, which are computed by a LSTM module. For value decomposition method, we choose QMIX as the baseline and a more powerful method, QPlex (J. Wang, Ren, Liu, Yu, & Zhang, 2021), for comparison. Qplex implements its value decomposition module based on the idea borrowed from dueling Q-learning (Sewak, 2019) and achieve state-of-the-art performance on SMAC environment.

Figure 2 shows the median win rate of different algorithms across all aforementioned super-hard scenarios. As we can see, TWG-Q achieves the best performance over all algorithms. Traditional communication based approaches perform very poor in such combat tasks, which is much worse than TWG-Q. IN *MMM2*, the win rate of TWG-Q is almost 90% while those of communication based methods are smaller than 20%. The gap of the win rate between TWG-Q and communication based methods is quite large over all maps. Compared with QMIX and Qplex, TWG-Q also performs better. For the first two super hard maps, i.e., *MMM2* and *27m_vs_30m*, TWG-Q improves the win rate by around 50% compared to QMIX. For the other two super hard maps in which QMIX and Qplex do not work well, TWG-Q still gets around a win rate of 50% and 20% respectively.

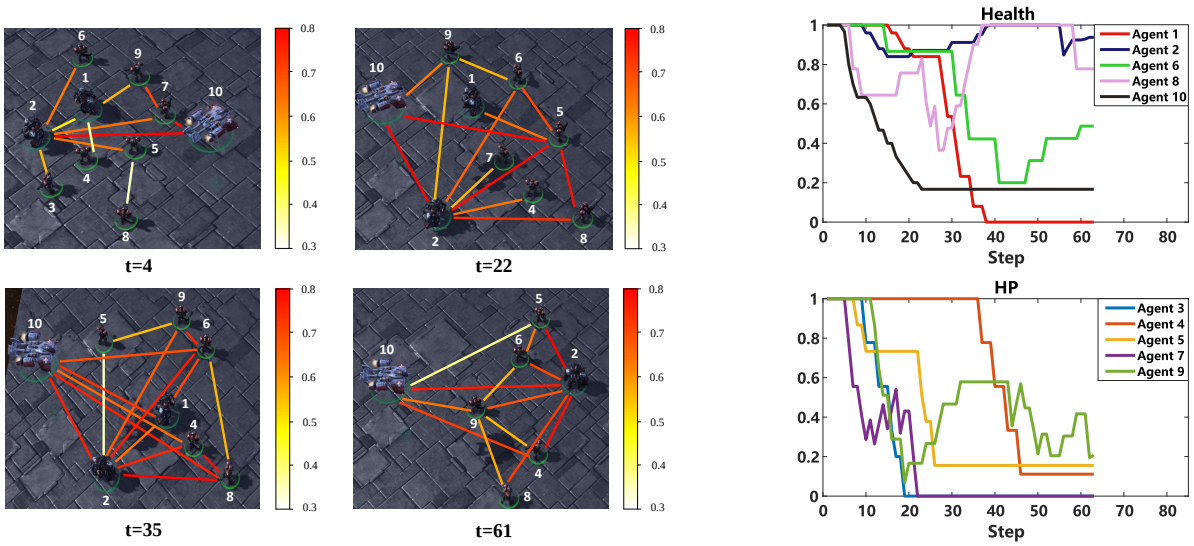
The scenario *1m2h_vs_8m* (1 Marauder and 2 Hellion vs 8 Marines) and *nuke_vs_cannon* (Ghost with Nuke vs Battle-Cruise with Cannon) are specially designed with temporal characteristics. In the scenario *1m2h_vs_8m*, the unit Hellion has a high attack power but needs much longer cooldown time than other attackers. In the scenario *nuke_vs_cannon*, the weapon of the unit ghost is Nuke which can make damages for enermies in a certain distance but takes a long time to be recharged. As we can see, TWG-Q performs especially better than other algorithms. Both value decomposition methods and other communication based methods get no more than 20% win rate while TWG-Q gets 50% win rate in scenario *1m2h_vs_8m*. In scenario *nuke_vs_cannon*, TWG-Q could get around 100%, which is 20% higher than QMIX. Other methods do not work at all. It is also interesting to see that TWG-Q learns some complicated cooperative behaviors for the two maps. The unit Marauder draw the enemy's fire while two Hellions could find good positions to make large damage for enemies.

Performance of Temporal Module

In this section, we conduct two experiments to evaluate the impact of the TCN module on our framework. In the first experiment, we compare the TCN module with two widely used sequence modules, i.e., GRU (Chung, Gulcehre, Cho, & Bengio, 2014) and the self-attention mechanism (Vaswani et al., 2017). We apply these modules in the weight learning mechanism to generate TC weight for the communication model. Experiments are conducted on special scenarios *1m2h_vs_8m* and *nuke_vs_cannon*. Both these scenarios require temporal cooperation between agents. Results are shown in Figure 3. We can observe that self-attention mechanism performs worst because it is not suitable to the dynamic changing temporal cooperative features. The performance of weight learning with GRU temporal module is better than that of self-attention model. Actually, GRU records all historical features and the large number of historical features may be detrimental to its performance. The TCN module achieves the best performance, which proves the effectiveness of our framework.

The second experiment concentrates on the impact of different sizes of receptive fields (*rf*s) on the weight learning mechanism. We set the size of receptive field to 4, 8, 16 and 32 respectively, and then evaluate the performance on two special scenarios, i.e., *1m2h_vs_8m* and *nuke_vs_cannon*. The win rate results are shown in Figure 3(c).

As we can see the receptive fields of size 16 and 8 perform the best in the two scenarios respectively. Thus it is not a good strategy to set the receptive field to the largest value or the smallest value. It is a hyper-parameter which depends on



(a) MMM2 Weighted Graphs.

(b) MMM2 Agent Health.

Figure 4: Results of weighted graph analysis.

concrete example. The best receptive field in the first scenario is smaller than that of the second scenario. The main reason is that the time interval of the attacking action for Hellion is 3 which is smaller than that of Nuke, i.e., 6. The two units are the most important units which could cause the biggest damage in the two scenarios respectively.

Analysis of Weighted Communications

We take one scenario MMM2 to analyse the variation of weighted communications, as well as some interesting cooperative behaviors. Figure 4(a) shows battle records at some representative steps $t = 4, 23, 35, 61$ respectively. In each figure, the importance of relations among agents are expressed in the form of heat map. Note that the relation between two agents is not shown in the figure if its weight is smaller than 0.3. A bigger weight on an edge represents a higher cooperation among two agents. Figure 4(b) shows the value of the health point (HP) for each agent. Medivac (Agent 10) is a unit which could suffer more damage and have the ability of healing others. The first important trick learned by TWG-Q is that Medivac flies in the front to draw the enemy's fire and then retreat to heal other fighting units. We can see that the health point of Medivac deceases quickly at the beginning in Figure 4(b). At $t = 4$, the weight between Medivac and the second Marauder (Agent 2) is the highest, which indicates that the cooperation of the two agents is important. This also explains the phenomenon that Medivac flies back to the second Marauder after this time step to complete the retreat. The second trick to win is to make the first Marauder (Agent 1) and all Marines move into the front to cause harm on enemies as much as possible, see records at $t = 22$ and $t = 35$. However, the price of causing so much damage is deaths of the first Marauder and some Marines (Agent 3 and 7). As shown in Figure 4(b), their HPs reduce to zero. At $t = 62$, the second Marauder leads all Marines to fight with enemies and finally win the game. In summary, weighted graphs show that the most important units in this battle are the Medivac and the second Marauder, as they have more connections than other units. The second Marauder leads Marines to fight, so their weights are quite high at $t = 62$. Further, if an unit needs to be healed, the weight of its connection with Medivac usually becomes high. Some high weights among Marines indicate that allied units can communicate with each other to complete the retreat action.

Conclusion

This paper proposes TWG-Q, a novel MARL framework with a weight learning mechanism based on temporal information. The proposed TWG-Q models cooperative relationships between agents as a complete graph, where all agents can communicate with each other. To distinguish valuable cooperative information, a temporal cooperative weight learning mechanism is designed to generated weights on the graph. Temporal cooperative features are used and dealt with properly to compute weights as accurate as possible. Especially for amounts of temporal cooperative features over a certain time range, a TCN module with the receptive field is designed to efficiently extract valuable temporal cooperative information. With the agent graph and the learned temporal cooperative weights, TWG-Q can perform efficient cooperative policy learning. Empirical results show that TWG-Q obtains the best performance on the challenging SMAC benchmark over state-of-the-art communication based MARL methods. Further, this study also provides an effective tool to analyze multi-agent cooperative behaviors among agents though the weighted cooperation graph. It would be interesting to investigate cooperative behaviors in other environments with our method.

References

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Böhmer, W., Kurin, V., & Whiteson, S. (2020). Deep coordination graphs. In *International conference on machine learning* (pp. 980–991).

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Foerster, J. N., Assael, Y. M., De Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. *arXiv preprint arXiv:1605.06676*.

Iqbal, S., & Sha, F. (2019). Actor-attention-critic for multi-agent reinforcement learning. In *International conference on machine learning* (pp. 2961–2970).

Jiang, J., Dun, C., Huang, T., & Lu, Z. (2020). Graph convolutional reinforcement learning. In *International conference on learning representations.*

Kim, D., Moon, S., Hostallero, D., Kang, W. J., Lee, T., Son, K., & Yi, Y. (2019). Learning to schedule communication in multi-agent reinforcement learning. *arXiv preprint arXiv:1902.01554*.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kong, X., Xin, B., Liu, F., & Wang, Y. (2017). Revisiting the master-slave architecture in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1712.07305*.

Lea, C., Flynn, M. D., Vidal, R., Reiter, A., & Hager, G. D. (2017). Temporal convolutional networks for action segmentation and detection. In *proceedings of the ieee conference on computer vision and pattern recognition* (pp. 156–165).

Li, S., Gupta, J. K., Morales, P., Allen, R., & Kochenderfer, M. J. (2021). Deep implicit coordination graphs for multi-agent reinforcement learning. In *Proceedings of the 20th international conference on autonomous agents and multiagent systems* (pp. 764–772).

Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., & Gao, Y. (2020). Multi-agent game abstraction via graph attention neural network. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 7211–7218).

Mao, H., Liu, W., Hao, J., Luo, J., Li, D., Zhang, Z., ... Xiao, Z. (2020). Neighborhood cognition consistent multi-agent reinforcement learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 7219–7226).

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Interspeech* (Vol. 2, pp. 1045–1048).

Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, *50*(9), 3826–3839.

OroojlooyJadid, A., & Hajinezhad, D. (2019). A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*.

Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., & Wang, J. (2017). Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, *2*, 2.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*.

Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., ... Whiteson, S. (2019). The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*.

Sewak, M. (2019). Deep q network (dqn), double dqn, and dueling dqn. In *Deep reinforcement learning* (pp. 95–108). Springer.

Sukhbaatar, S., Szlam, A., & Fergus, R. (2016). Learning multiagent communication with backpropagation. *arXiv preprint arXiv:1605.07736*.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V. F., Jaderberg, M., ... others (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Aamas* (pp. 2085–2087).

Tacchetti, A., Song, H. F., Mediano, P. A., Zambaldi, V., Rabinowitz, N. C., Graepel, T., ... Battaglia, P. W. (2018). Relational forward models for multi-agent learning. *arXiv preprint arXiv:1809.11044*.

Tung, T.-Y., Pujol, J. R., Kobus, S., & Gunduz, D. (2021). A joint learning and communication framework for multi-agent reinforcement learning over noisy channels. *arXiv preprint arXiv:2101.10369*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, J., Ren, Z., Liu, T., Yu, Y., & Zhang, C. (2021). Qplex: Duplex dueling multi-agent q-learning. In *International conference on learning representations.*

Wang, W. Z., Shih, A., Xie, A., & Sadigh, D. (2022). Influencing towards stable multi-agent interactions. In *Conference on robot learning* (pp. 1132–1143).

Wunder, M., Littman, M. L., & Babes, M. (2010). Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *Icml.*

Yang, Y., Hao, J., Liao, B., Shao, K., Chen, G., Liu, W., & Tang, H. (2020). Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*.

Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., ... others (2018). Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*.

Zatsarynna, O., Abu Farha, Y., & Gall, J. (2021). Multi-modal temporal convolutional network for anticipating actions in egocentric videos. In *Cvpr* (pp. 2249–2258).