ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9413716

GRAPHCOMM: A GRAPH NEURAL NETWORK BASED METHOD FOR MULTI-AGENT REINFORCEMENT LEARNING

Siqi Shen[†], *Yongquan Fu*[‡*], *Huayou Su*[‡], *Hengyue Pan*[‡], *Peng Qiao* [‡], *Yong Dou*[‡], *Cheng Wang*[†*]

[†] Fujian Key Lab of Sensing and Computing for Smart Cities, School of Informatics, Xiamen University, China
[‡] School of Computer, National University of Defense Technology, China

ABSTRACT

The communication among agents is important for Multi-Agent Reinforcement Learning (MARL). In this work, we propose GraphComm, a method makes use of the relationships among agents for MARL communication. GraphComm takes the *explicit* relations (e.g., agent types), which can be provided through some knowledge background, into account to better model the relationships among agents. Besides explicit relations, GraphComm considers *implicit* relations, which are formed by agent interactions. GraphComm use Graph Neural Networks (GNNs) to model the relational information, and use GNNs to assist the learning of agent communication. We show that GraphComm can obtain better results than state-of-the-art methods on the challenging StarCraft II unit micromanagement tasks through extensive experimental evaluation.

1. INTRODUCTION

Multi-Agent Reinforcement Learning (MARL) problems, where a group of agents with partial observable environment must cooperate to achieve their common goal, have receive much attention from the research community [1].

The communication among agents provides additional information by message passing. It enable the agents to obtain better information around them. Researchers have proposed various methods to learn to communicate. The attentional communication model (ATOC) [2] enable agents to select and communicate with each other; target communication [3] allows agent transfer messages to targeted agents. Albeit these communication methods can be efficient at some tasks, these methods do not fully take into account the rich *relational information* that naturally exists among agents. Thus, they cannot capture agent relationships well, which leads to sub-optimal performance for challenging tasks [4].

In Multi-Agent System (MAS), there exist many types of relationships (e.g., healer-warrior). Communication through these relationships can be effective. We propose to consider

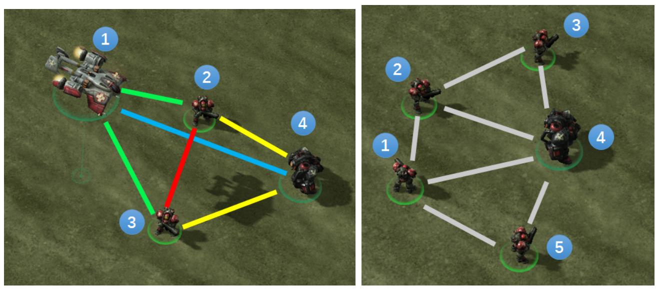

Fig. 1. The Explicit (Left) and the Implicit (Right) Relationship Graphs in StarCraft II scenarios. The color of an edge indicates a distinct relationship.

both *explicit* and *implicit* relations simultaneously among agents. *Explicit* relationships are explicitly described (or predefined) relationships, which are obtained from some knowledge background; they precisely define the relationships among agents. For example, the alliance-enemy, the hider-seeker, and the speaker-listener relationship. However, using explicit relationships alone are not enough to capture various mutual information in MAS. There are many *implicit* relationships that are not explicitly described but exist in MAS. Implicit relationships are formed through agent interactions. For example, an agent could collide/chase another agent, stand in front/back of another agent.

Figure 1 depicts two examples of relationships formed in StarCraft II games. The left figure shows explicit relations induced by agent types. There are 4 agents with 3 types: 1 Medivac, 1 Marauder, and 2 Marines. Agents of different types are related to other agent types differently, and there are 4 types of explicit relationships (e.g., Medivac-Marine, Medivac-Marauder) in the figure. The right part of Figure 1 demonstrates the implicit relationships formed according to distances between agents. Nearby agents are connected by gray edges. The implicit relationships are dynamic. For example, during a StarCraft combat, a Medivac agent may heal or protect a Marine agent.

In this work, we propose GraphComm. It uses Graph Neural Networks (GNNs) to model the relational information, and use GNNs to assist the learning of agent communication through back-propagation. Through extensive experiments,

*indicates corresponding authors. Email: siqishen@xmu.edu.cn. This work was supported by the National Natural Science Foundation of China (No.U1605254, 61602500, 61972409, 61502500, 61702529, 61702530)

we show that GraphComm can perform significantly better than state-of-the-art methods on the challenging StarCraft II unit micromanagement tasks by virtue of using the *explicit* and *implicit* relationships for communication.

2. RELATED WORK

Multi-Agent communication has received much attention from the research community [5, 6, 7]. Graphs have been used as a communication mechanism for agent cooperation. ATOC [8] uses the attention mechanism to select and communicate with others. DGN [9] and COMA-GAT [10] use the Graph Attention Network (GAT) [11] as a communication mechanism to enlarge the observation of agents. G2ANet [12] uses hard-attention to filter irrelevant information and soft-attention to focus on relevant information. DCG [13] uses coordination graphs that coordinate the actions of agents through message passing on graphs. Most of the method use proximity-based relational graphs for message-passing during agent execution. Different from them, GraphComm agents use both the *explicit* and *implicit* relationships to communicate during execution.

Action-critic MARL methods [14, 15] use the signals from critics to guide the optimization of actors. Value-based approaches, such as VDN [16], QMIX [17] factorize the value function Q_{tot} into multiple local value function Q_i. Our work is orthogonal to these approaches.

3. BACKGROUND

The MARL scenarios we consider is cooperative, it can be modeled as Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) [18]. For n agents, the Dec-POMDP is defined as G.

$$G = \langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, P, r, \{\mathcal{O}_i\}_{i=1}^n, \{\sigma_i\}_{i=1}^n, n, \gamma\rangle.$$

The set of states are denoted by $\mathcal{S}$. The set of actions of agent i is represented by $\mathcal{A}_i$, in this work, only discrete actions are considered. $\boldsymbol{a}^t \in \mathcal{A} := \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ denotes the joint action of all agents. At a discrete time step t and state s^t, after the joint action is issued, the next state $s^{t+1} \in \mathcal{S}$ of the environment is drawn from the transition function $s^{t+1} \sim P(\cdot|s^t, \boldsymbol{a}^t)$. In cooperative MARL, agents must coordinate and cooperate to achieve their common goal. All the agents will receive a collaborative reward r^t after the state transition happens. Each agent can only observe a part of the environment $o_i^t \in \mathcal{O}_i$ which is drawn from $o_i^t \sim \sigma^i(\cdot|s^t)$. $\gamma \in [0, 1)$ is the discount parameter. The agents seek to optimize the value function $Q_{tot}(s, a) = \mathbb{E}_{s^{0:\infty}, a^{0:\infty}}[\sum_{t=0}^{\infty} \gamma^t r^t | s^0 = s, a^0 = a]$. In this work, Each agent can communicate with other agents through predefined channel which is formed base on relationships.

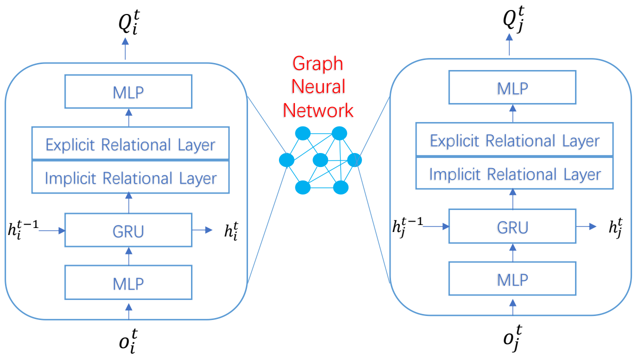

Fig. 2. The architecture of GraphComm.

4. THE GRAPHCOMM METHOD

The relational information can be used in MARL communication to enable better cooperative agent behaviors. The relationships among agents are modeled as explicit and implicit relational graphs. GraphComm uses GNN to model the *explicit* and *implicit* relational information among agents to exchange messages, thus benefiting the agent execution.

4.1. Architecture

GraphComm architecture is shown in Figure 2. The neural architecture of the agent consists of a multi perception layer (MLP) layer that encodes the local observation (o_i^t), a Gate Recurrent Layer (GRU) [19] layer, which encodes the history of local action-observation. Further, the agents exchange their messages with Graph Neural Network through the Relational Graph Module (RGM). It consists of the Implicit Relational Layer and the Explicit Relational Layer. In the end, a fully-connected layer is used to output Q_i^t. All the agents share the same set of parameters.

The local value function Q_i^t are fed into a mixing network (i.e., [17]) to compute a global value function Q_{tot}^t for centralized training. During execution, agent i selects its action u_i greedy according to $Q_i(o_i^t, u)$ without the mixing network.

4.2. Relational Graph Module

GraphComm models a Multi-Agent System (MAS) as a graph. An agent is a node of the graph; the edge of the graph is formed based on the relational information among agents. For example, an edge between two agents can be created if they are within each other's observation. The Relational Graph Module (RGM) uses the explicit and the implicit relational layer to model explicit and implicit relationships, respectively.

4.2.1. *Explicit Relations*

We consider explicit relations that are predefined in the form of a relational graph $\mathcal{G}$ which is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$. $\mathcal{V}$ is the set of nodes in the graph. $v_i \in \mathcal{V}$ is a node in the graph. It could be agents or other entities in the environment. $\mathcal{E}$ is the set of edges. An edge e_{ij} from v_i to v_j indicates a relationship between them. $\mathcal{R}$ is the set of types of relationships. There exists an mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$ which maps an edge e_{ij} to its relationship type $r \in \mathcal{R}$.

The relational graph could be provided by background knowledge or learned from previous MARL. We consider explicit relationships that are consistent over the course of MARL. For example, as it is shown in Figure 1 (Left), the Medivac agent 1 can heal the Marine agent 2. e_{12} from 1 to 2 represents the can-heal relationship. and e_{21} from 2 to 1 indicates the can-be-healed relationship. The healing relationships are consistent over the course of the StarCraft II scenario.

4.2.2. *Implicit Relations*

We consider implicit relations that are formed through agent interaction. They are not provided to MARL beforehand, and the relationships emerge during the course of MARL.

For example, the proximity relationship could be a type of implicit relationship if the locations of agents are dynamic or unknown in advance. For proximity relationships, we measure the distance between the positions of agents in Euclidean space. If the distance between two agents is smaller than a predefined threshold, then a proximity relation exists between the two agents. Other types of implicit relationships may emerge. For example, a warrior agent may *follow* or *cover* another agent.

We build an implicit relationship graph base on the proximity relational graph. That is, two agents are considered having an implicit relationship only if they are close enough. In this work, we consider a dynamic MARL that agents can disappear in the middle of the tasks. If an agent disappears, then all the edges connected with the agents are removed.

4.2.3. *Explicit Relational Layer*

For explicit relational graphs, GraphComm requires that the knowledge across edge type (i.e., relationship) be shared. We find that the Relation Graph Convolution Network (RGCN), which uses a distinct shared function to compute for each edge type, satisfies our requirement. Thus, it is adopted to implement the explicit relational layer.

Given set of input to the explicit relational layer: $\mathbf{h}^{(l)} = h_1^{(l)}, h_2^{(l)}, ...h_n^{(l)}, h_i^{(l)} \in \mathbb{R}^F$, where $h_i^{(l)}$ is the hidden states of agent i in the lth layer. For example, it could be the local observation o_i^t of agent i, or the transformed hidden variable of agent i from previous layer. This layer outputs a set of new agent features $\mathbf{h}^{(l+1)} = h_1^{(l+1)}, h_2^{(l+1)}, ...h_n^{(l+1)}$. $h_i^{(l+1)} \in \mathbb{R}^{F'}$. The transformation of $h_i^{(l)}$ to $h_i^{(l+1)}$ is defined as follows.

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{ir}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right) \qquad (1)$$

$r \in \mathcal{R}$ represents the type of an explicit relationship, $\mathcal{R}$ is the set of types of relationships. $\mathcal{N}_i^r$ is the set of neighbors which connect i with relation r, $c_{ir} = |\mathcal{N}_i^r|$, $W_r^{(l)}$ is a type-specific transformation matrix. It is shared across relationships of the same type r. $W_0^{(l)}$ transforms the hidden features of agent i. Overall, the explicit relational layer aggregates neighboring agents' hidden features into a normalized sum which consider different explicit relations.

4.2.4. *Implicit Relational Layer*

To model the implicit relational graph, GraphComm uses the Graph Attention Network (GAT) [11] to implement the implicit relational layer. GAT aggregates nearby agents' information via attending over neighboring agents' features. The output of a GAT layer is defined as follows.

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i}\alpha_{ij}h_j^{(l)}\right) \qquad (2)$$

where $h_j^{(l)}$ is the output from previous layer, $h_i^{(l+1)}$ is the output of the GAT layer. α_{ij} is the attention weight between h_i and h_j. W is a vector. Further, we use multi-head attention to jointly attend different representation subspace to extract useful relational representations.

4.3. Training

The loss of s the TD error $\mathcal{L}_{mse}$ which is defined as follows.

$$\mathcal{L}_{mse} = \sum_{i=1}^{b}(y_{tot}^i - Q_{tot}(s^i, a^i; \theta))^2 \qquad (3)$$

where b is the batch size, a^i is the joint action, $y_{tot}^i = r + \gamma max_a Q_{tot}(s^{i+1}, a; \theta^-)$, θ^- are the parameters of the target network. The training process is the same as DDQN [20].

5. EVALUATION

We conduct experiments on the StarCraft II Multi-Agent Challenge benchmark (SMAC) [21]. The results show that GraphComm can obtain *better* results than **8** state-of-the-art approaches by virtue of using explicit and implicit relationships.

	GraphComm	QMIX	QAtten	VDN	QTRAN	DGN	IQL	COMA	COMA-GAT
MMM2	0.83	0.65	0.49	0.05	0.01	0.13	0.03	0.00	0.00
8m_vs_9m	0.90	0.71	0.84	0.83	0.47	0.39	0.24	0.03	0.03
3s5z_vs_3s6z	0.08	0.04	0.03	0.01	0.00	0.00	0.00	0.00	0.00
6h_vs_8z	0.09	0.04	0.04	0.00	0.00	0.00	0.00	0.00	0.00
3s5z	0.94	0.90	0.89	0.48	0.03	0.03	0.06	0.01	0.01
MMM	0.99	0.97	0.99	0.98	0.45	0.78	0.63	0.39	0.25

Table 1. The Test Win Rate of all methods.

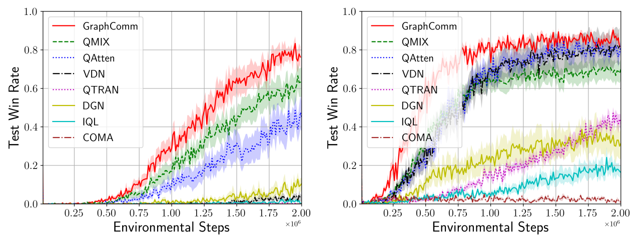

Fig. 3. The test won rate for the MMM2 (Left) and the 8m_vs_9m (Right) scenarios.

5.1. StarCraft II benchmark

In a SMAC scenario, a group of agents controlled by learning policies fights against enemies controlled by in-game bots. We choose 6 StarCraft II scenarios consisting of different types and number of agents, and require different strategies to win. For example, there are 3 types of agents in the MMM2 and MMM scenarios: Marine, Marauder, and Medivac. Each agent has a circular view around it and can receive local observation around it. It can move and attack enemies or heal teammates. In each episode, if the agents eliminated all the enemies within a limited time, the game episode is counted as won, otherwise loss.

5.2. Metric, Methods, and Configuration

The training procedure is paused after every 10,000 environment steps, and then 32 test episodes are conducted to evaluate the performance in terms of *Test Win Rate*. Test Win Rate is the percentage of games won among the test episodes.

The work used for comparison are: QMIX [17], QAtten [22], VDN [16], QTRAN [4], DGN [9], IQL [23], COMA [15], COMA-GAT [10]. The input and output dimensions of the relational graph module are both 64. The number of attention heads is 4. To strike a balance between computational requirement and statistical significance, each experiment is repeated 10 times with different seeds. Otherwise specified, the configuration of GraphComm is the same as QMIX [17].

We provide GraphComm with explicit relational graphs based on the type of agents. For agents with the type A (e.g., Marine), edges with the relation *AA* is created among these agents. For an agent with type A and another agent with type B, an edge with the relation *AB* is created between the two agents. For scenarios with homogeneous type of agents (8m_vs_9m, 6h_vs_8z), we set agent 1 as type A, whereas other agents as type B. Then, the explicit relational graphs are created as previously described. The implicit relational graph is built based on agents' relative distance. If the distance between two agents is smaller than a threshold, then an edge (implicit relationships) is created between them. Examples of the relational graph are shown in Figure 1.

5.3. Results

The final mean results (maximal mean across the testing episodes within the last $250k$ steps of training) are shown in Table 1, and part of the training results are depicted in Figure 3 (with 95% confidence interval). GraphComm achieves the *best* results for all the scenarios.

For the MMM2 scenario, the performance of GraphComm is consistently better than others. The win rate of GraphComm is 0.83, whereas the second and the third-best result is achieved by QMIX (0.65) and QAtten (0.49), respectively. For the 8m_vs_9m scenario, depicted in Figure 3 (Right), GraphComm is the best performing method, it can obtain a win rate of 0.90, while the second-best algorithm (QAtten) is 0.84. For the 3s5z_vs_3s6z, 6h_vs_8z, 3s5z, MMM scenarios, whose results are shown in Table 1, GraphComm can obtain the best performance. The results indicate that considering both the explicit and implicit relationship for MARL communication can improve the performance of MARL.

6. CONCLUSION

In this work, we propose GraphComm, a method uses both the *explicit* and the *implicit* relationships among the agents for MARL communication. Agents exchange messages among other through the communication channel based on relationships. And GraphComm models the relationships using Graph Neural Networks. Through extensive experiments, we show that GraphComm can obtain better results than state-of-the-art methods.

7. REFERENCES

[1] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor, "Is multiagent deep reinforcement learning the answer or the question? A brief survey," *CoRR*, vol. abs/1810.05587, 2018.

[2] Jiechuan Jiang and Zongqing Lu, "Learning attentional communication for multi-agent cooperation," in *NeurIPS*, 2018, pp. 7265–7275.

[3] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau, "Tarmac: Targeted multi-agent communication," in *ICML*, 2019, pp. 1538–1546.

[4] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi, "QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *ICML*, 2019, vol. 97, pp. 5887–5896.

[5] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *NeurIPS*, 2016, pp. 2137–2145.

[6] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang, "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games," *CoRR*, vol. abs/1703.10069, 2017.

[7] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus, "Learning multiagent communication with backpropagation," in *NeurIPS*, 2016, pp. 2244–2252.

[8] Jiechuan Jiang and Zongqing Lu, "Learning attentional communication for multi-agent cooperation," in *NeurIPS*, 2018, pp. 7265–7275.

[9] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu, "Graph convolutional reinforcement learning," in *ICLR*, 2020.

[10] Jianyu Su, Stephen C. Adams, and Peter A. Beling, "Counterfactual multi-agent reinforcement learning with graph convolution communication," *CoRR*, vol. abs/2004.00470, 2020.

[11] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio, "Graph attention networks," *ICLR*, 2018.

[12] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao, "Multi-agent game abstraction via graph attention neural network," in *AAAI*, 2020, pp. 7211–7218.

[13] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson, "Deep coordination graphs," *ICML*, 2020.

[14] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NeurIPS*, 2017, pp. 6379–6390.

[15] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI*, 2018, pp. 2974–2982.

[16] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *AAMAS*, 2018, pp. 2085–2087.

[17] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," in *ICML*, 2018, vol. 80, pp. 4292–4301.

[18] Frans A. Oliehoek and Christopher Amato, *A Concise Introduction to Decentralized POMDPs*, Springer Briefs in Intelligent Systems. 2016.

[19] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *EMNLP*, 2014.

[20] Hado van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double q-learning," in *AAAI*, 2016, pp. 2094–2100.

[21] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson, "The starcraft multi-agent challenge," in *AAMAS*, 2019, pp. 2186–2188.

[22] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang, "Qatten: A general framework for cooperative multiagent reinforcement learning," *CoRR*, vol. abs/2002.03939, 2020.

[23] Ming Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *ICML*, Paul E. Utgoff, Ed., 1993, pp. 330–337.