QRELATION: AN AGENT RELATION-BASED APPROACH FOR MULTI-AGENT REINFORCEMENT LEARNING VALUE FUNCTION FACTORIZATION

Siqi Shen[†], *Jun Liu*[†], *Mengwei Qiu*[†], *Weiquan Liu*[†], *Cheng Wang*[†*], *Yongquan Fu*[‡*], *Qinglin Wang*[‡], *Peng Qiao* [‡]

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

ABSTRACT

The Centralized Training with Decentralized Execution paradigm (CTDE), which trains policies centrally with additional information, is important for Multi-Agent Reinforcement Learning (MARL). For CTDE, value function factorization methods make use of state during training and factorize the value function into multiple local value functions for decentralized execution. These approaches do not fully consider the relational information among agents, resulting in sub-optimal models for complex tasks. To remedy this issue, we propose QRelation which is a graph neural network approach for value function factorization. It considers both the *static* relations (e.g., agent types) and *dynamic* relations (e.g., close-by). We show that QRelation can obtain better results than state-of-the-art methods on challenging StarCraft II benchmarks.

1. INTRODUCTION

Many of the multi-agent systems (MAS) problems, such as robot control [1], and vehicle driving [2] can be modeled as cooperative Multi-Agent Reinforcement Learning (MARL) problems where a group of agents must cooperate to achieve their common goal. MARL has attracted great research interest due to its social and economic impact [3]. MARL is hard to learn due to the scalability and the partial-observability issues [4]. To address these issues, many approaches adopt the Centralized Training with Decentralized Execution paradigm (CTDE) [5]. To learn decentralized policies, CTDE uses additional information, such as reward and state, during training. CTDE has attracted recent attention from the MARL community [6]. However, many challenges about how to exploit CTDE remain.

Policy-based and value factorization based methods exploit CTDE by making use of states during training. Policy-based methods [7] learn a value function Q_{tot} to guide the optimization of decentralized policies. However, they are sample-inefficient due to their on-policy nature. Value factorization approaches [6, 8] factorize Q_{tot} into multiple local value function Q_i; and each agent selects its action greedily with respect to Q_i. The use of value factorization is motivated by exploiting the *independence* among agents in CTDE. These methods do not take into account the rich *relations* that naturally exists among agents, leading to sub-optimal performance for complex tasks [8].

Relations are the way to which two agents are connected. In MAS, there exist many types of relationships (e.g., healer-warrior), which could be used in CTDE. We propose to consider both *static* and *dynamic* relations simultaneously among agents. *Static* relationships are explicitly described and consistent relationships. For example, the alliance-enemy, the predator-prey relationships. Moreover, there are many *dynamic* relationships that formed through agent interactions. For example, an agent could collide/chase another agent, stand in front/back of another agent.

In this work, we propose QRelation, which fully makes use of the relational information of agents for value function factorization. QRelation models the MAS as a graph, where each agent represents a node, and an edge connecting two nodes represents a relationship. QRelation models static relationships, such as connections among agents of various types, through the use of an *static* relational layer. And it uses the *dynamic* relational layer on graphs to model the dynamic relationships. QRelation is flexible enough to allow for multiple edges between a pair of agents, as the two agents could have numerous relationships. To obtain consistent agent behaviors, we adopt a relational regularizer that restricts the learned relational representations (i.e., attention weights) from changing too frequently.

We conduct large-scale experiments to study the performance of QRelation on the StarCraft II MARL tasks. The results show that QRelation can perform better than state-of-the-art approaches through using both static and dynamic agent-based relationships.

2. RELATED WORK

Researchers have proposed various MARL communication methods [3, 9, 10] and actor-critic methods [11, 12]. Value factorization approaches are widely adopted in MARL.

[*]indicates corresponding authors. Email: siqishen@xmu.edu.cn; yongquanf@nudt.edu.cn; cwang@xmu.edu.cn. The source code is available at https://github.com/xmu-rl-3dv/QRelation

VDN [13] factorizes the value function Q_{tot} as a sum of all the value functions of agents Q_i. During execution, each agent acts greedily according to the Q_i. VDN supports the additive relationships among Q_i and Q_{tot}. QMIX [6] supports monotonic relationships among Q_i and Q_{tot}, and Weighted QMIX[14] improves it through weighting optimal actions. QTRAN [8] supports more general factorization by transforming the value function into a VDN-factorized function with linear constraints. QTRAN++ [15] improves QTRAN by introducing more training constraints and well-designed neural networks. Qatten [16] adopts the attention mechanism to focus on certain agents/scenarios when factorizing value function. QPlex [17] uses dueling network to factorize Q_{tot}. DMIX [18] integrates distributional RL with QMIX. As a value factorization method, QRelation considers the relationships among agents and models the static and dynamic relational information through GNN and the attention mechanism.

3. THE QRelation METHOD

The key to QRelation is the insight that besides the state and observation, the relational information can be used in CTDE to obtain better cooperative agent behaviors. QRelation uses graphs to model the *static* and *dynamic* relational information among agents to build a comprehensive value function, thus benefiting the agent execution.

3.1. Dec-POMDPs

The cooperative Multi-Agent Reinforcement Learning (MARL) scenarios can be modeled as Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) [19]. $G = \langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, P, r, \{\mathcal{O}_i\}_{i=1}^n, \{\sigma_i\}_{i=1}^n, n, \gamma \rangle$. $\mathcal{S}$ is the set of states; $\mathcal{A}_i$ is the set of discrete actions for agent i. A joint action of all agents is defined as $\boldsymbol{a}^t \in \mathcal{A} := \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$. 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 $s^{t+1} \sim P(\cdot|s^t, \boldsymbol{a}^t)$. The agents receive a shared 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.

3.2. Overview

The architecture of QRelation is depicted in Figure 1. QRelation factorizes the value function Q_{tot} into individual local value function Q_i. Agent i selects its action u_i greedily according to $Q_i(o_i^t, u)$, without communication with each other. The neural architecture of the agent consists of a fully-connected layer that encodes the local observation, a Gate Recurrent Layer (GRU) [20] layer, which encodes the history of local action-observation, and a fully-connected layer which outputs Q_i. All the agents share the same set of parameters.

QRelation takes local value function Q_i^t, observations o_i^t, and state s^t as input, and train a centralized critic which outputs the value function Q_{tot}^t for the joint action of all agents. The relationships among agents are modeled as static and dynamic relational graphs. QRelation uses the Relational Module (RM), which will be described later, to model the relationships. o_i^t is fed into RM to obtain a key variable k_i^t, which encodes both the agent's local observation and its relationships. Then, the keys $k_1^t, k_2^t, ..., k_n^t$ are fed into the attention layer along with Q_i and s^t to obtain Q_{tot}^t.

3.3. Relational Module

The Relational Module (RM) uses the static and the dynamic relational layer to model static and dynamic relationships, respectively. In default, the local observations of agents are fed into the dynamic relational layer, and then its output is used as input for the static relational layer. Note that, the order of the static and the dynamic relational layer can be switched depending on the application scenarios, and multiple RM can be stacked together.

3.3.1. Static Relations

QRelation assumes that static relations are predefined in the form of a relational graph $\mathcal{G}$ which is defined as $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{T})$. $\mathcal{N}$ is the set of agents. $n_i \in \mathcal{N}$ is an agent in the graph. $\mathcal{E}$ is the set of edges. An edge e_{ij} denotes a static relationship between agent n_i and agent n_j. $\mathcal{T}$ is the set of types of static relationships. Each edge is associated with one type $t \in \mathcal{T}$. To model the static relations t between agents, we use the Relational Graph Convolution Network (RGCN) [21] as the implementation of static relation module to capture agents' static relationships. Specifically, RGCN uses a distinct shared weight matrix for each type t, thus it can capture each t well.

3.3.2. Dynamic Relations

The dynamic relations among agents emerge through agent interaction, and change over the course of MARL. For example, proximity-based relationships are a type of dynamic relationships. If two agents are close, then a proximity-based dynamic relation exists between the two agents. There are other types of dynamic relationships. For example, an agent may be listening to another agent.

QRelation uses the Graph Attention Network (GAT) [22] as the implementation of the dynamic relational module. First, for each agent, GAT calculates its neighboring agents' attention weights with respect to it. Second, GAT multiplies the weights with neighboring agents' features. In the end, the other agents' features are represented as a sum over the products. Given the key k_i (e.g., $w_k o_i$), the query q (e.g.,

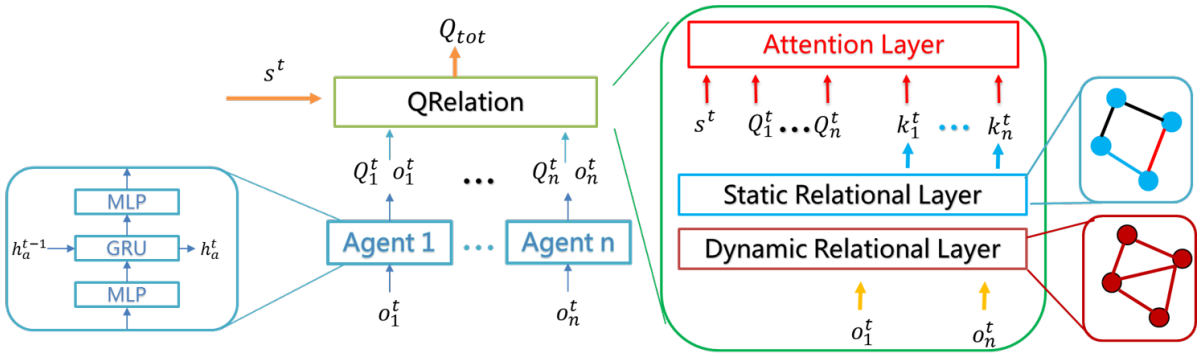

Fig. 1. The architecture of QRelation.

$w_q o_j$), the *attention coefficients* α_i of the dynamic relational module for agent i are defined as follows.

$$\alpha_i = \frac{\exp(f(q, k_i))}{\sum_j \exp(f(q, k_j))}. \quad (1)$$

where f is a user-defined function.

3.4. Attention Layer and Value Function

After the Relational Graph Module, the local observation $o_1^t, o_2^t ..., o_n^t$ are transformed into a set of hidden variables $h = k_1^t, k_2^t, ...k_n^t$. k_i^t contains the aggregated and focused information for agent i from its neighboring agents. Thus, k_i^t contains more comprehensive features than o_i^t. Similar to [11, 16], QRelation adopts the self-attention mechanism to estimate the impact of each agent i for the team. The attention coefficient w_i^t of agent i is calculated as follows.

$$w_i^t = \frac{\exp(w_k^a k_i^t \cdot w_s^a s^t)}{\sum_{j=1}^n \exp(w_k^a k_j^t \cdot w_s^a s^t)} \quad (2)$$

where s^t is the state, W_s and W_k are weight matrices, and $\cdot$ is the dot-product operation.

The value function Q_{tot}^t is calculated as the sum between the weighted sum of agent's Q values and a state value $V(s)$. $Q_{tot}^t = \sum_{i=1}^n w_i^t Q_i^t + V(s^t)$, where $V(s^t) \in \mathbb{R}$ is a state value function, which is obtained by feeding s^t through 2 fully-connected layers with Relu non-linearity.

3.5. Relation Regularization and Loss

In QRelation , the dynamic relational graph changes according to the interaction of the agents. As many cooperative MARL problems need consistent agent behaviors, we want dynamic relationships in QRelation to be stable for a while, even when the surrounding agents change.

The attention weights α of the dynamic relational layer are the learned relational representations. Inspired by [23], QRelation uses a relational graph regularization $\mathcal{L}_{gnn}$ to regulate the weights of the relational graph, to ensure that the relationships at time step $t+1$ do not differ too much from last time step t. It is defined as follows.

$$\mathcal{L}_{gnn} = \sum_{t=1}^{T} \sum_{i=1}^{n} KL(\alpha_i^t || \alpha_i^{t+1}) \quad (3)$$

where α_i^t are the attention weights distribution of agent i at time t used in Formula 1, and $KL(||)$ is the KL divergence distance.

The loss of QRelation consists of the TD error, the relational graph regularizer $\mathcal{L}_{gnn}$. It is defined as follows.

$$\mathcal{L} = \mathcal{L}_{mse} + \lambda_1 \mathcal{L}_{gnn} \quad (4)$$

where $\mathcal{L}_{mse}$ is the TD error. QRelation is trained end-to-end to minimize $\mathcal{L}$ in the same manner as DDQN.

4. EVALUATION

We evaluate QRelation using the StarCraft II Multi-Agent Challenge benchmark (SMAC) [24]. The results show that QRelation can obtain *better* results than **7** state-of-the-art approaches by virtue of using static and dynamic relationships. Further, we conduct an ablation study which shows that combining static and dynamic relationships are necessary, and the relational regularizer L_{gnn} is useful.

4.1. StarCraft II benchmark

In the StarCraft MARL benchmark, several combat agents controlled by learning policies fight against enemy bots. We choose 2 super hard and 2 easy StarCraft II scenarios consisting of different types and number of agents. They require different strategies to win. As an example, in the MMM2 scenario, there are agents types: Medivac, Marauder, and Marine. If the controlled agents defeat all the enemy bots in limited time, the episode is counted as won, otherwise lose.

4.2. Metric, Methods, and Configuration

The training procedure of all the methods is paused after every 10,000 environment steps, and then we conduct 32 test

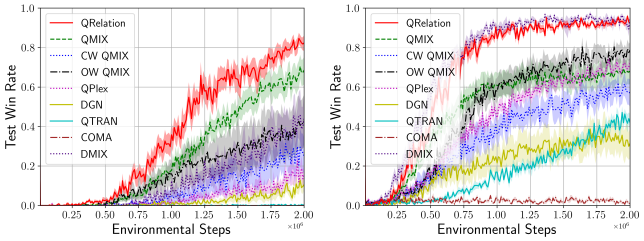

Fig. 2. The test won rate for the super-hard scenarios: MMM2 (Left) and 8m_vs_9m (Right).

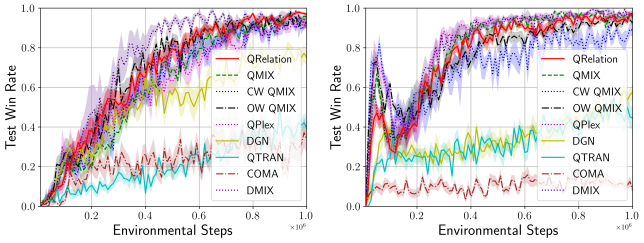

Fig. 3. The test won rate for the easy scenarios: MMM (Left) and 2s3z (Right).

episodes to evaluate their performance in terms of *Test Win Rate*. It is the percentage of won test episodes.

The state-of-the-art methods used for comparison are: QMIX [6], CW QMIX [14], OW QMIX [14], QPlex [17], DGN [23], QTRAN [8], COMA [7], and DMIX [18]. The input and output dimensions of the relational module are both 64, and each attention layer consists 4 attention heads. Each experiment is repeated 5 times with independent runs. Unless otherwise specified, the configuration of QRelation is the same as QMIX [6]. The static relational information is obtained from the type of StarCraft II agents (e.g., Marine and Medivac). And we consider distance-based dynamic relational information only. If two agents are close, then a dynamic relationship exists between them.

4.3. Results

The Test Win Rate of the scenarios are depicted in Figure 2 and Figure 3 (with 95% confidence interval). QRelation achieves the *best* results for the two super-hard scenarios (MMM2 and 8m_vs_9m) and obtains close-to-optimal results in the two easy scenarios (MMM and 2s3z). For the MMM2 scenario, the performance of QRelation is consistently better than others. The win rate of QRelation is 0.85, whereas the second and is achieved by QMIX (0.65) respectively. For the 8m_vs_9m scenario, depicted in Figure 2 (Right), QRelation and DMIX are the best performing methods, they can obtain a win rate of 0.97. Note that the agent used in DMIX is distributional and rnn-based while other methods use the same rnn-based agent. For the two easy scenarios, as shown in Figure 3, QRelation can obtain close-to-optimal performance

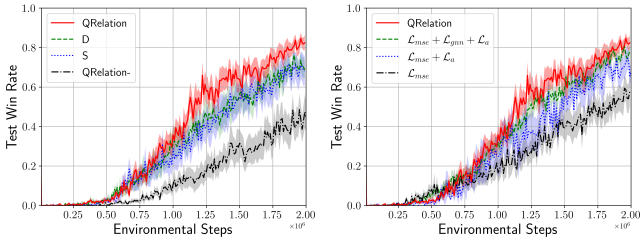

Fig. 4. Impact of different relationship (Left) and loss (Right).

as other methods.

To study the impact of static and dynamic relations, we consider the following ablations: **S**, **D**, and QRelation-. **S** is a variation of QRelation which only considers static relations, whereas **D** is an variant considers dynamic relations only. **S** and **D** use one layer of static/dynamic relational layer, respectively. QRelation- is an variation which do not use any relationships at all. As it is depicted in Figure 4 (Left). QRelation obtains better performance than all these variations through using both the static and dynamic relations. And all the methods considering relationships obtain better results than the one which does not use any relationships (i.e., QRelation-).

To study the impact of relational regularizer, we introduce another relational regularizer L_a. It considers each agent's relative importance to the team should be stable, instead of changing radically. L_a restricts the attention coefficient w_i^t in the attention layer (defined in Formula 2) from changing too frequently. The loss L_a is defined as $\mathcal{L}_a = \sum_{t=1}^{T} KL(\mathbf{w^t}||\mathbf{w^{t+1}})$. Besides, we consider QRelation's variant L_{mse} which uses the TD loss only, variant $L_{mse} + L_a$ which uses the TD loss and L_a, and variant $L_{mse} + L_a + L_{gnn}$ considering all the three losses. QRelation can be viewed as $L_{mse} + L_{gnn}$. As it is shown in Figure 4 (Right), considering TD loss only (L_{mse}) achieves the poorest performance, whereas the two relational regularizers (L_a and L_{gnn})can improve the performances. Through using the relational regularizer L_{gnn}, QRelation achieves the best performance, and using the three losses L_{mse}, L_a, L_{gnn} together do not lead to better results.

5. CONCLUSION

QRelation is a relation-based value function factorization approach for multi-agent reinforcement learning. It uses graph neural networks to model both the static and the dynamic relational information, respectively. Thus, QRelation captures the relation among agents well. We show that QRelation can perform better than state-of-the-art methods by modeling static and dynamic relationships through experiments on StarCraft II benchmarks.

6. REFERENCES

[1] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib, "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes," in *AAAI*, 2012.

[2] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, 07 2012.

[3] 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.

[4] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *ICML*, 2017, pp. 2681–2690.

[5] Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos A. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *J. Artif. Intell. Res.*, vol. 32, pp. 289–353, 2008.

[6] 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.

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

[8] 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.

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

[10] Siqi Shen, Yongquan Fu, Huayou Su, Hengyue Pan, Peng Qiao, Yong Dou, and Cheng Wang, "Graphcomm: A graph neural network based method for multi-agent reinforcement learning," in *ICASSP*, 2021.

[11] Shariq Iqbal and Fei Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *ICML*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds., 2019.

[12] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang, "Off-policy multi-agent decomposed policy gradients," *ICLR*, 2020.

[13] 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.

[14] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson, "Weighted QMIX: expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *NeurIPS*, 2020.

[15] Kyunghwan Son, Sungsoo Ahn, Roben Delos Reyes, Jinwoo Shin, and Yung Yi, "Qtran++: Improved value transformation for cooperative multi-agent reinforcement learning," 2020.

[16] 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.

[17] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang, "QPLEX: duplex dueling multi-agent q-learning," in *ICLR*, 2021.

[18] Wei-Fang Sun, Cheng-Kuang Lee, and Chun-Yi Lee, "DFAC framework: Factorizing the value function via quantile mixture for multi-agent distributional q-learning," in *ICML*, 2021.

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

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

[21] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling, "Modeling relational data with graph convolutional networks," 2018, ESWC.

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

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

[24] 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.