Quantum Multi-Agent Reinforcement Learning via Variational Quantum Circuit Design

‡Won Joon Yun, §Yunseok Kwak, †Jae Pyoung Kim, ‡Hyunhee Cho,
§Soyi Jung, †Jihong Park, and †Joongheon Kim

†School of Electrical Engineering, Korea University, Seoul, Republic of Korea
‡School of Electrical Engineering, Sungkyunkwan University, Suwon, Republic of Korea
§School of Information Technology, Deakin University, Geelong, Victoria, Australia

Abstract—In recent years, quantum computing (QC) has been getting a lot of attention from industry and academia. Especially, among various QC research topics, variational quantum circuit (VQC) enables quantum deep reinforcement learning (QRL). Many studies of QRL have shown that the QRL is superior to the classical reinforcement learning (RL) methods under the constraints of the number of training parameters. This paper extends and demonstrates the QRL to quantum multi-agent RL (QMARL). However, the extension of QRL to QMARL is not straightforward due to the challenge of the noise intermediate-scale quantum (NISQ) and the non-stationary properties in classical multi-agent RL (MARL). Therefore, this paper proposes the centralized training and decentralized execution (CTDE) QMARL framework by designing novel VQCs for the framework to cope with these issues. To corroborate the QMARL framework, this paper conducts the QMARL demonstration in a single-hop environment where edge agents offload packets to clouds. The extensive demonstration shows that the proposed QMARL framework enhances 57.7% of total reward than classical frameworks.

Index Terms—Quantum deep learning, Multi-agent reinforcement learning, Quantum computing

I. INTRODUCTION

The recent advances in computing hardware and deep learning algorithms have spurred the ground-breaking developments in distributed learning and multi-agent reinforcement learning (MARL) [1]. The forthcoming innovations in quantum computing hardware and algorithms will accelerate or even revolutionize this trend [2], motivating this research on quantum MARL (QMARL). Indeed, quantum algorithms have huge potential in reducing model parameters without compromising accuracy by taking advantage of quantum entanglement [3]. A remarkable example is the variational quantum circuit (VQC) architecture, also known as a quantum neural network (QNN) [4], [5], which integrates a quantum circuit into a classical deep neural network. The resultant hybrid quantum-classical model enables quantum reinforcement learning (QRL) that is on par with classical reinforcement learning (QRL) methods under the constraints of the number of training parameters [4]. Mathematically, there are two ways to describe a qubit state: |

\[ |0\rangle, |1\rangle = \frac{1}{\sqrt{2}} (|0\rangle + \beta |1\rangle), \quad \beta e^{i\varphi} = \cos(\delta/2)|0\rangle + e^{i\varphi} \sin(\delta/2)|1\rangle, \forall \delta, \varphi \in [-\pi, \pi]. \]

II. QUANTUM COMPUTING AND CIRCUIT

A. Quantum Computing in a Nutshell

Quantum computing utilizes a qubit as the basic unit of computation. The qubit represents a quantum superposition state between two basis states, which denoted as $|0\rangle$ and $|1\rangle$. Mathematically, there are two ways to describe a qubit state:
Experience

optimization, approximation, and classification. As shown in features on VQC, which will are for QMARL.

which changes the sign of the second qubit if the first qubit is generates quantum entanglement between those qubits. Among qubit according to the signal of several control qubits, which multiple qubits, called controlled rotation gates. They act on a qubit. In contrast, there are quantum gates which operate on 

operator gates by rotating quantum circuit. The next step is into corresponding qubit states, which can be treated in the and in this step, a classical input information is encoded where $\otimes$

process can be formulated as follows:

$$\mu_{\pi}(s_t|o_t) = \text{softmax}(f(o_t; \theta)),$$

where softmax$(x) \triangleq \left[ \frac{e^{x_1}}{\sum_{i=1}^{N} e^{x_i}}, \cdots, \frac{e^{x_N}}{\sum_{i=1}^{N} e^{x_i}} \right]$. At time $t$, the actor policy of the $n$-th agent makes action decision with the given observation $o_{n}^{t}$, which is denoted as $\mu_{\pi_{n}}(u_{n}^{t} | o_{n}^{t})$. Note that $\theta_{n}$ denotes parameters of $n$-th actor. Then, the action $u_{n}^{t}$ is computed as follows:

$$u_{n}^{t} = \arg \max_{u_{n}} \pi_{\theta_{n}}(u_{n}^{t} | o_{n}^{t}).$$

2) Quantum Centralized Critic: We adopt the centralized critic for CTDE as a state-value function. At time $t$, the parameterized critic estimates the discounted returns given $s_t$ as follows:

$$V^{\psi}(s_t) = f(s_t; \psi) \approx E\left[ \sum_{t'=t}^{T} \gamma^{t'-t} \cdot r(s_{t'}, u_{t'}) | s_t = s \right],$$

where $\gamma$, $T$, $u_{t}$, and $r(s_{t'}, u_{t'})$ stand for a discounted factor $\gamma \in [0, 1)$, an episode length, the actions of all agents, and reward functions, respectively. In addition, $\psi$ presents trainable parameters of a critic. Note that $s_t$ is the ground truth at $t$. Note that the state encoding is used as shown in green box in Fig. 1 because the state size is larger than the size in observation.
Algorithm 1: CTDE-based QMARL Training

1. Initialize the parameters of actor-critic networks and the replay buffer, \( \Theta \triangleq \{ \Theta^A, \ldots, \Theta^N \}, \psi, \phi, \mathcal{D} = \{ \} \);
2. repeat
3. \( t = 0, s_0 = \) initial state;
4. while \( s_t \neq \) terminal and \( t < \) episode limit do
5. \( \quad \) for each agent \( n \) do
6. \( \quad \) \( u^n_t \) \( \leftarrow \) \( \text{sample from} \) \( u^n_t \);
7. \( \quad \) end
8. \( \quad \) Get reward \( r_t \) and next state and observations \( s_{t+1} \);
9. \( \quad \) Get \( \mathcal{D} = \mathcal{D} \cup \{ s_t, u_t, r_t, s_{t+1}, o_{t+1} \} \);
10. \( \quad t = t + 1 \); step = step + 1;
11. end
12. for each timestep \( t \) in each episode in batch \( \mathcal{D} \) do
13. \( \quad \) Compute \( y_t \);
14. \( \quad \) end
15. \( \quad \) Calculate \( \nabla_{\Theta, \psi} J, \nabla_{\phi} J, \) and update \( \Theta, \psi \);
16. \( \quad \) if target update period then
17. \( \quad \quad \) Update the target network, \( \phi \leftarrow \psi \);
18. \( \quad \) end
19. until obtaining optimal policies;

TABLE I: The MDP of a single-hop offloading environment

| Observation | \( q^n_t = \{ q^n_{t,1}, \ldots, q^n_{t,K} \} \cup \{ s_{t,K} \} \) |
|---|---|
| Action | \( u^n_t \in \mathcal{A} = \mathbb{I} \times \mathcal{P} \) |
| \( \in \mathcal{P} \) | \( \mathcal{P} = \{ \min_{t,1}, \ldots, K \} \) |
| State | \( s_t = \cup_{n=1}^N \{ o^n_t \} \) |
| Reward | \( r(s_t, u_t) \in \{ \) in (1) |

TABLE II: The experiment parameters.

| Parameters | Values |
|---|---|
| The numbers of clouds and edge agents (\( K, N \)) | 2, 4 |
| The packet amount space (\( \mathcal{P} \)) | \( \{0.1, 0.2\} \) |
| The hyper-parameters of environment (\( \psi_T, w_R \)) | \( \{0.3, 4\} \) |
| Transmitted packets from the cloud (\( b_c^k \)) | 0.3 |
| The capacity of queue (\( q_{\text{max}} \)) | 1 |
| Optimizer | Adam |
| The number of gates in \( U_{\text{gate}} \) | 50 |
| The number of qubits of actor/critic | 4 |
| Learning rate of actor/critic | \( 1 \times 10^{-4}, 1 \times 10^{-5} \) |

IV. EXPERIMENTS AND DEMONSTRATIONS

A. Single-Hop Offloading Environment

The environment used in this paper consists of \( K \) clouds and \( N \) edges. The clouds and edges have queues \( q^c_t \) and \( q^e_t \) that temporally store packets. All edge agents offload their packets to clouds. The queue dynamics are as follows:

\[
q^c_{t+1} = \text{clip}(q^c_{t+1} - u^c_t - b^c_t, 0, q_{\text{max}}),
\]

where the superscript \( i \in \{ c, e \} \) identifies the cloud and an edge device. The terms \( u^c_t \) and \( b^c_t \) imply the total transmitting packet size and the packet arrival of \( k \)-th cloud or \( n \)-th edge, respectively. Note that \( u^c_t = 0 \) and \( b^c_t = 0 \) if \( n = 0 \).

\[ q^c_{t+1} = \text{clip}(q^c_{t+1} - u^c_t - b^c_t, 0, q_{\text{max}}), \]

where the superscript \( i \in \{ c, e \} \) identifies the cloud and an edge device. The terms \( u^c_t \) and \( b^c_t \) imply the total transmitting packet size and the packet arrival of \( k \)-th cloud or \( n \)-th edge, respectively. Note that \( u^c_t = 0 \) and \( b^c_t = 0 \) if \( n = 0 \).
classical CTDE MARL where the number of parameters is more than 40K. The simulation parameter settings are listed in Table II. We use python libraries (e.g., torchquantum and pytorch) for deploying VQCs and DL methods, which support GPU acceleration [12]. In addition, all experiments are conducted on AMD Ryzen™ Threadripper™ 1950x and NVIDIA RTX 3090. We have confirmed that the training time of QMARL for 1,000 epochs is not expensive (≈ 35 minutes).

D. Evaluation Results

1) Reward Convergence: Fig. 3 presents the demonstration results. As shown in Fig. 3(a), the reward of QMARL frameworks is around -3.0 for Proposed and -16.6 for Comp1, whereas the classical MARL frameworks record -22.5 for Comp2 and -2.8 for Comp3, respectively. We calculate the achievable utility as min-max normalization with the average returns of random walks. Note that the random walk records -33.2 on average. The achievable utility of QMARL frameworks is 90.9% for Proposed and 49.8% for Comp1. However, the classical MARL frameworks achieve 33.2% for Comp2 and 91.5% for Comp3. In summary, the proposed QMARL outperforms hybrid QMARL and classical MARL under the constraint of the number of trainable parameters.

2) Performance: The average queue states of edges/clouds and clouds are 0.460 for Proposed, 0.480 for Comp1, 0.510 for Comp2, and 0.453 for Comp3, respectively. The ratio of the number of empty queue events records in a high order of Comp2, Comp1, Proposed, and Comp3. However, the overflowed queue is low with the order of Proposed, Comp3, Comp2, and Comp1. According to Fig. 3(a–d), the QMARL framework outperforms both classical and hybrid quantum-classical MARL frameworks under the constraints of the number of trainable parameters.

E. Demonstration

Due to high network latency of utilizing quantum clouds, we conduct demonstration on simulation. Fig. 4 shows the visualization of the workflow of our QMARL framework. The superpositioned qubit states (i.e., magnitude and, phase vector) are expressed as 4 × 4 heatmap in hue-lightness-saturation color system. We provide source codes1 including QMARL, the single-hop environment, and the simulator.

V. CONCLUDING REMARKS AND FUTURE WORK

This paper introduces quantum computing concepts to MARL, i.e., QMARL. To resolve the challenge of QMARL, we adopt VQC with state encoding and the concept of CTDE. From the single-hop environment, we verify the superiority of QMARL corresponding to the number of trainable parameters and the feasibility of QMARL. As a future work direction, the implementation of QMARL to the quantum cloud (e.g., IBM quantum, Xanadu, or IonQ) should be interest because the impact of noise is considerable on quantum computing.

Acknowledgement. This research was supported by the National Research Foundation of Korea (2022R1A2C2004869 and 2021R1A4A1030775). W. J. Yun and Y. Kwak contributed equally to this work. S. Jung, J. Park, and J. Kim are corresponding authors.

REFERENCES

[1] J. Park, S. Samarakoon, A. Elgabli, J. Kim, M. Bennis, S.-L. Kim, and M. Debbah, “Communication-efficient and distributed learning over wireless networks: Principles and applications,” Proceedings of the IEEE, vol. 109, no. 5, pp. 796–819, 2021.
[2] M. Schuld and N. Killoran, “Is quantum advantage the right goal for quantum machine learning?” CoRR, vol. abs/2203.01340, 2022.
[3] S. Oh, J. Choi, and J. Kim, “A tutorial on quantum convolutional neural networks (QCNN),” in Proc. of IEEE Int’l Conf. on ICT Convergence (ICTC), October 2020.
[4] Z. Hong, J. Wang, X. Qu, X. Zhu, J. Lin, and J. Xiao, “Quantum convolutional neural network on protein distance prediction,” in Proc. IEEE Int’l Joint Conf. on Neural Networks (IJCNN), July 2021.
[5] Y. Kwak, W. J. Yun, S. Jung, and J. Kim, “Quantum neural networks: Concepts, applications, and challenges,” in Proc. IEEE Int’l Conf. on Ubiquitous and Future Networks (ICUFN), August 2021.
[6] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-X. Chen, X. Ma, and H.-S. Goan, “Variational quantum circuits for deep reinforcement learning,” IEEE Access, vol. 8, pp. 141 007–141 024, 2020.
[7] Y. Kwak, W. J. Yun, S. Jung, J.-K. Kim, and J. Kim, “Introduction to quantum reinforcement learning: Theory and PennyLane-based implementation,” in Proc. IEEE Int’l Conf. on ICT Convergence (ICTC), October 2021.
[8] G. Carleo, I. Cirac, K. Cramer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, “Machine learning and the physical sciences,” Reviews of Modern Physics, vol. 91, no. 4, p. 045002, 2019.
[9] P. W. Shor, “Scheme for reducing decoherence in quantum computer memory,” Physical Review A, vol. 52, no. 4, p. R2493, 1995.
[10] J. Biamonte, “Universal variational quantum computation,” Physical Review A, vol. 103, no. 3, p. L030401, 2021.
[11] N. Wiebe, A. Kapoor, and K. M. Svore, “Quantum deep learning,” CoRR, vol. abs/1412.3489, 2014.
[12] H. Wang, Y. Ding, J. Gu, Z. Li, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, “QuantumNAS: Noise-adaptive search for robust quantum circuits,” in Proc. IEEE Int’l Symposium on High-Performance Computer Architecture (HPCA), April 2022.

1 https://github.com/WonJoon-Yun/Quantum-Multi-Agent-Reinforcement-Learning