Multi-agent Quantum Reinforcement Learning for Digital Twin Placement in 6G Multi-tier Systems

Shehbaz Tariq[1], Muhammad Shohibul Ulum[1], Abdurrahman Wachid Shaffar[1], Wook Park[1], Sunghwan Kim[2], and Hyundong Shin[1(✉)]

[1] Department of Electronics and Information Convergence Engineering, Kyung Hee University, Seoul, Republic of Korea
hshin@khu.ac.kr

[2] Department of Electrical, Electronic and Computer Engineering, University of Ulsan, Ulsan, Republic of Korea

Abstract. The digital twins (DTs) represent critical components for the simulation, analysis, and optimization of physical systems, with significant implications for efficiency and cost management in 6G network applications. The introduction of multi-tier computing has the potential to enable a more streamlined integration of DTs in 6G networks by offering services at the edge network level. However, this integration brings about various complexities related to the placement and maintenance of DTs within edge networks, thus increasing the processing latency. To address these problems, we investigate the application of quantum computing, which exploits quantum principles such as superposition and entanglement, and offers potential resolutions for these computational dilemmas. In this paper, we formulate the DT placement problem to incorporate variational quantum circuits for learning agents within a multi-agent reinforcement learning framework. In particular, quantum multi-agent reinforcement learning is proposed to establish an optimal policy for associating DTs with edge networks. This approach aims to reduce latency while adhering to the computational resource limitations of the edge server. Simulation results illustrate the proficiency and robustness of quantum multi-agent actor-critic networks in acquiring a policy that ameliorates the reward function, hence decreasing latency while adhering to the optimization constraints. This study contributes to the evolving field of quantum computing applications in multi-tier

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (NRF-2019R1A2C2007037, NRF-2022R1A4A3033401, NRF-2023R1A2C1003546) and by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2023–2021-0–02046) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation). This research is partially funded by the BK21 FOUR program of National Research Foundation of Korea.

N.-S. Vo et al. (Eds.): INISCOM 2024, LNICST 595, pp. 73–92, 2024.
https://doi.org/10.1007/978-3-031-67357-3_6

environments and provides methodological insights for optimizing DT deployment in 6G networks.

Keywords: Digital twin · Multi-tier computing · Quantum reinforcement learning · Digital twin placement

1 Introduction

Intelligent ubiquitous connectivity and unprecedented communication performance of sixth-generation (6G) networks have enabled extreme ultra-reliable low-latency communications (xURLLC) for critical applications such as autonomous cars and telemedicine, massive machine-type communications (mMTC) for smart IoT devices, and enhanced mobile broadband plus (eMBB-Plus) for high-bandwidth applications such as virtual and augmented reality [9,10,21,26]. Such ultra-massive connectivity for billions of smart devices, ultra-high precision for reliability and latency, and ultra-broadband for perpetual data traffic provide high-quality user experiences for emerging digital applications. However, it requires significant advancements in conventional computing and communication technologies, efficient resource management and optimization, and supportive regulations and business models [7,17].

Recently, the concept of DT has gained significant attention in communication networks to improve efficiency, reduce costs, and improve decision-making [15]. Virtual representation of physical reality in DTs aids in simulating, analyzing, and optimizing the behavior of the physical system. However, DT modeling requires abundant amounts of data from smart IoT devices as well as edge computing resources to process the data [14]. Contrarily, the emergence of multi-tier computing can potentially assist in DT realization in a 6G network. Integrating DT and multi-tier computing systems in 6G networks can provide a more efficient and effective way to manage and optimize the performance of various physical systems digitally. Moreover, by executing computation at the intelligent edges using mobile edge computing (MEC), network latency is reduced, ultimately improving real-time decision-making performance [36]. Therefore, MEC-assisted DT technology can reliably simulate and precisely analyze the system's behavior, leading to increased prediction accuracy and effective optimization of the system [18].

On the one hand, MEC allows for computation and storage tasks to be performed at the edge of the network, closer to end users, enabling low latency and high bandwidth services, such as streaming real-time video and audio, augmented reality, and autonomous vehicles [19]. MEC servers are typically located in base stations or other network edge points, allowing them to process data from nearby devices and respond faster to requests or queries. This reduces the amount of data that must be transmitted to and from the central server, which reduces latency and improves overall network performance [1]. Additionally, MEC can improve network security by processing sensitive data locally, thereby culminating in the need to transmit it to the cloud [2]. However, the convergence of DTs

and edge computing poses several challenges related to the placement and maintenance of DTs in edge networks. One of the main challenges is the dynamic nature of network states, such as channel states and available computational resources, which can affect the performance of DTs and the underlying applications. To address this, the mapping relations between end devices and DTs in the edge server must be carefully designed to ensure optimal performance [8,32].

To ensure the optimal performance of DTs in edge networks, it is crucial to address the dynamic nature of network states and the challenges posed by the convergence of DTs and edge computing. This also includes careful design of the mapping relations between the end devices and DTs on the edge server. In the realm of quantum computing, the exponential computational power offered by quantum computers holds significant potential with respect to DT placement. Quantum computers utilize qubits, which can simultaneously exist in multiple states, and employ entanglement, a fundamental principle of quantum mechanics, to produce instantaneous correlations between qubits. This unique capability enables quantum computers to perform parallel computations and process large amounts of data simultaneously. Recent empirical evidence, including experiments on programmable superconducting processors and programmable photonic processors, has showcased the quantum supremacy of these devices, demonstrating their ability to tackle problems beyond the reach of classical computers [22,27].

Quantum machine learning (QML) is a rapidly developing field that seeks to revolutionize conventional machine learning by harnessing the computational power of quantum computers. As quantum computers continue to advance and more data becomes available, QML is expected to play an increasingly significant role in various applications, currently unaddressable with conventional computing machines [11,28]. In particular, in situations involving large datasets and intricate patterns, QML models have shown promising potential to outperform classical models [12,13].

In this context, the benefits of quantum neural networks have been reported for sequence learning tasks by exploring the influence of quantum contextuality on the expressiveness of a Gaussian-based model [3]. The findings highlight the distinctive expressive capabilities of quantum neural networks compared to classical neural network models and demonstrate that quantum contextuality can enhance the efficiency of sequence learning tasks while providing insights into optimal conditions for effectiveness.

Furthermore, quantum computing has been applied to various optimization problems, including network optimization, resource allocation, and scheduling [5,33]. Optimization problems, which involve identifying the optimal solution within given constraints, pose challenges even for classical computing systems. However, quantum algorithms, such as quantum approximate optimization algorithm (QAOA) and quantum annealing-based methods, have demonstrated significant effectiveness in addressing combinatorial optimization problems and surpassing classical algorithms in terms of efficiency. The aforementioned problem is known to be NP-hard, indicating its computational complexity for classical com-

puting systems. However, quantum computing offers promising solutions through algorithms such as quantum-inspired reinforcement learning (QRL), and Quantum Multi-Agent Actor-Critic Networks are effective in finding optimal resource management policies [24,34]. Alternate approaches such as QAOA use a series of quantum gates to approximate the solutions of combinatorial optimization problems. On the other hand, quantum annealing achieves optimization by gradually reducing the temperature of a quantum system toward its ground state. Both methods aim to provide the most favorable solutions to optimization problems. These quantum-based methods have seen successful application across a broad range of optimization problems, spanning from scheduling and resource allocation to financial trading. Their versatility and performance further underscore the immense potential of quantum computing in solving complex optimization problems [4,35].

2 Related Works

QRL has the potential to revolutionize artificial intelligence by enabling faster and more efficient learning algorithms. In [29], the authors investigate using a quantum communication channel with the environment to accelerate the learning process of a reinforcement learning (RL) agent. By combining classical communication with this quantum scenario, the authors demonstrate their capacity to evaluate and optimize the obtained improvement. The experimental system includes a compact, fully adjustable nanophotonic integrated processor. Another study [13] introduces a novel quantum approach to deep reinforcement learning, emphasizing exploiting the potential for quadratic quantum speedups to manage large state and action spaces effectively. The authors proposed deep energy-based models, which drew inspiration from statistical physics. These models are more efficient than conventional deep reinforcement learning architectures such as deep Q-networks. Using quantum approximate sampling algorithms overcomes the challenge of sampling from associated probability distributions. The findings of this study contribute to the development of deep reinforcement learning methods capable of effectively navigating environments with vast state and action spaces. In [34], the authors employ quantum-inspired reinforcement learning to address the excessive task-offloading problem in vehicular networks. The quantum intelligent edge offloading strategy for resource-constrained vehicles effectively manages processing delay, energy consumption, and cost, thereby increasing the quality of experience of users. In [25], the authors proposed a quantum multi-agent actor-critic network for facilitating collaboration among multiple unmanned aerial vehicles Fig. 1. intermediatescale quantum (NISQ) computing to further equip the multi-agent reinforcement learning technique in addressing data-intensive wireless service quality issues and training speed Fig. 4.

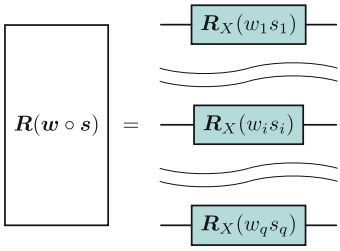

Fig. 1. Data Encoding: The variational quantum circuit (VQC) structure that illustrates how the input data is parameterized and encoded for processing in QRL. It uses a vector $\boldsymbol{w}$ to parameterize the input via Hadamard product with the state vector $\boldsymbol{s}$. This resultant vector is then encoded into quantum states using rotation gates, as depicted in the figure. Here, q is the dimension of the state vector $\boldsymbol{s}$.

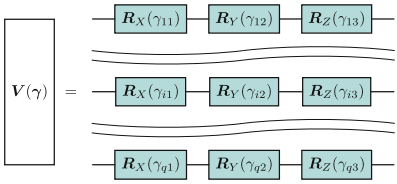

Fig. 2. Quantum Data Processing: A quantum circuit is applied for data processing. Here, $\boldsymbol{V}(\boldsymbol{\gamma}) = \bigotimes_i \boldsymbol{R}_Z(\gamma_{i_3})\boldsymbol{R}_Y(\gamma_{i_2})\boldsymbol{R}_X(\gamma_{i_1})$ represents the sequence of rotation operations. Here, q is the dimension of the state vector $\boldsymbol{s}$.

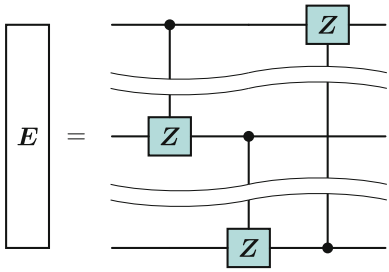

Fig. 3. Entanglement Layer: The application of controlled-Z gates in a cyclic pattern to introduce entanglement to the circuit. This step is can be represented as $\boldsymbol{E} = \prod_{i=N}^{1} \boldsymbol{C_Z}^{i}_{i+1}$, where $i+1$ is taken modulo N.

3 Quantum Reinforcement Learning

3.1 Reinforcement Learning

RL involves an agent that learns to behave in an environment by receiving rewards for its actions and considering the state of the environment. RL can be defined as a Markov decision process [31], which includes a set of states $\mathcal{S}$, actions $\mathcal{A}$, and reward $\mathcal{R}$. The agent interacts with the environment in discrete

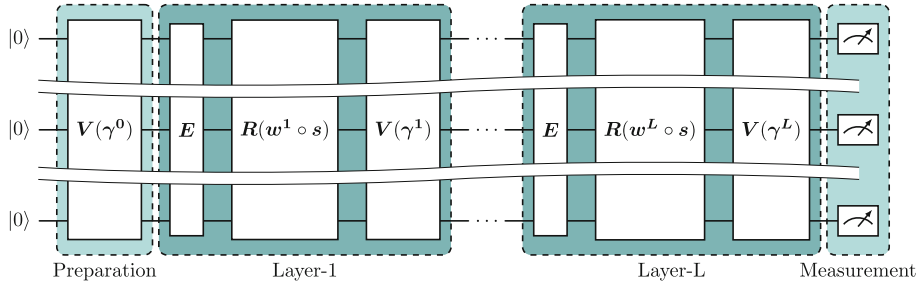

Fig. 4. variational quantum circuit (VQC) Structure for Actor and Critic Network: Circuit schematic for the multi-layered unitary transformation in (VQC) used for both actor and critic networks. The structure expressed as $\boldsymbol{U}(\boldsymbol{\theta}, \boldsymbol{s}) = \left(\prod_{l=L}^{1} \boldsymbol{V}(\boldsymbol{\gamma}^l)\boldsymbol{R}(\boldsymbol{w}^l \circ \boldsymbol{s})\boldsymbol{E}\right)\boldsymbol{V}(\boldsymbol{\gamma}^{\mathbf{0}})$, demonstrates the integral role of entanglement and rotation operations at each layer l in the data reuploading process.

time steps, observing a state $s_t \in \mathcal{S}$ at each step. Using the observed state, the agent selects an action $a_t \in \mathcal{A}$ according to a deterministic or stochastic policy. The action of the agent influences the environment, transitioning it to a new state s_{t+1}. Based on the triplet (s_t, a_t, s_{t+1}), the agent receives a reward r_t Fig. 2.

The agent aims to select an action that maximizes the cumulative reward at time t, which is calculated as

$$R_t = \sum_{\tau=0}^{\infty} \gamma^{\tau} r_{t+\tau}, \tag{1}$$

where $0 < \gamma \leq 1$ is a discount factor that determines the impact of future rewards. The agent finds the best action through a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$. The agent is guided by an action value function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ such that the policy is updated by maximizing the action value function. To help stabilize the learning process of the actor and critic network, the target actor and target critic network are introduced. The target critic and target actor network have the same model as the critic and target network; however, the target network parameter is not updated during the training phase. After each training phase is complete, the target network parameter is updated by copying the parameter of the trained networks.

For each time step, the agent observes the state s_t. Based on the observed state, the agent takes action a_t using the policy π. The agent receives a reward r_t and observes the next state s_{t+1}. The agent stores (s_t, a_t, r_t, s_{t+1}) in the replay memory. In the training phase, the agent sampled the data from the replay memory and update the critic network with the loss function [16],

$$L_c = \mathbb{E}[((r_t + \gamma Q(s_{t+1}, a_{t+1})) - Q(s_t, a_t))^2]. \tag{2}$$

The actor is updated with the loss function,

$$L_a = -\mathbb{E}[Q(s_t, \boldsymbol{\pi}(s_t)]. \tag{3}$$

3.2 Quantum Computing

Quantum computing uses a quantum mechanical framework to perform computation where the information is stored in a quantum state, and the computing process is done by the dynamics of a quantum system Fig. 3. The basic unit of quantum computing is a quantum state that is called qubit, which can be represented by a vector in a bi-dimensional Hilbert space $\mathbb{C}^2$. The common notation used is the braket notation such as [23],

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{4}$$

The states $|0\rangle$ and $|1\rangle$ are usually called the computational basis. Following quantum mechanical framework, the qubit can be in a superposition state,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{5}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. The quantum system dynamic is governed by a Hamiltonian $\boldsymbol{H}$ via Schrodinger's equation,

$$\iota\hbar \frac{\partial|\psi(t)\rangle}{\partial t} = \boldsymbol{H}|\psi(t)\rangle, \tag{6}$$

where $\hbar$ is the Planck constant. For a constant $\boldsymbol{H}$, the Schrodinger's equation has solution

$$|\psi(t)\rangle = \exp\left(-\iota \boldsymbol{H} t/\hbar\right)|\psi(0)\rangle. \tag{7}$$

The operator $\boldsymbol{U} = \exp\left(-\iota \boldsymbol{H} t/\hbar\right)$ is called a unitary operator, which has an inverse that enables the reversible operation over a quantum system. The common single qubit unitary operator are $\mathbf{I} = |0\rangle\langle 0| + |1\rangle\langle 1|$, $\mathbf{X} = |0\rangle\langle 1| + |1\rangle\langle 0|$, $\mathbf{Z} = |0\rangle\langle 0| - |1\rangle\langle 1|$, and $\mathbf{Y} = \iota\mathbf{X}\mathbf{Z}$.

The quantum system can also be built up using multiple qubits. The state of this composite system is described using the tensor product. Let $\{|k_i\rangle\}$ is basis states of the ith system, the composite system can be written as,

$$|\psi\rangle = \sum_{k_1, k_2, \ldots, k_n \in \{0,1\}} \alpha_{k_1}\alpha_{k_2}\cdots\alpha_{k_n}|k_1\rangle|k_2\rangle\cdots|k_n\rangle, \tag{8}$$

where $|k_1\rangle|k_2\rangle\cdots|k_n\rangle = |k_1\rangle \otimes |k_2\rangle \otimes \cdots \otimes |k_n\rangle$ and $\sum_{k_1, k_2, \ldots, k_n \in \{0,1\}} |\alpha_{k_1}\alpha_{k_2}\cdots\alpha_{k_n}|^2 = 1$.

In multi qubits system, entanglement phenomena can occur where one cannot write the composite system as the tensor product of each individual system. For instance, in two qubits system, the entangled state $|\psi\rangle$ cannot be expressed as

the tensor product of each individual system $|\psi_1\rangle|\psi_2\rangle$. Such an entangled state can be generated using an entangling operator such as CNOT that is given as

$$\mathbf{C_X} = |0\rangle\langle 0|\otimes\mathbf{I} + |1\rangle\langle 1|\otimes\mathbf{X}. \tag{9}$$

CNOT operator can transform the non-entangled state $|\psi\rangle = (|00\rangle + |10\rangle)/\sqrt{2}$ (we don't write the subscript that denotes in which system the state belongs to simplify the expression) to an entangled state $\boldsymbol{C_X}|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$.

After the information in the qubits is processed, the measurement is applied to extract the information from the quantum state via an observable. In a quantum system, an observable is represented as a Hermitian operator. For a single qubit case, the common observable is the $\boldsymbol{Z}$ operator that corresponds to computational basis measurement that is given by $\mathbf{M_1} = |0\rangle\langle 0|$ and $\mathbf{M_{-1}} = |1\rangle\langle 1|$. This measurement gives the outcome 1 and -1 when measuring state $|\psi\rangle$ with probability,

$$p(i) = \langle\psi|\mathbf{M_i}|\psi\rangle, \tag{10}$$

where $i \in \{1, -1\}$ and $\langle 0|$ denotes the conjugate transpose of $|0\rangle$ which can be written via $\dagger$ operation such as $|\cdot\rangle^\dagger = \langle\cdot|$. The expected value of the $\boldsymbol{Z}$ operator can hence be calculated as,

$$\langle\psi|\mathbf{Z}|\psi\rangle = \sum_i i\langle\psi|\mathbf{M_i}|\psi\rangle = \sum_i ip(i). \tag{11}$$

3.3 Variational Quantum Circuit

The sequence of unitary operators makes a quantum circuit. Instead of using a fixed circuit, trainable parameters can be introduced to the quantum circuit to make variational quantum circuit (VQC). VQC enables the quantum circuit to learn to perform various tasks, including optimization and approximation [6]. One of the key aspects of the VQC is to design the structure of the unitary operators commonly known as ansatz. While the ansatz can vary based on the task but the structure typically consists of the common parameterized unitary operators,

$$\mathbf{R_O}(\theta) = \exp\left(-\iota\mathbf{O}\frac{\theta}{2}\right), \tag{12}$$

where $\mathbf{O} \in \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. The two-qubits entangling gate used is the controlled-Z gate that is represented as,

$$\mathbf{C}_{\mathbf{Z}_i^j} = |0\rangle\langle 0|^{(j)}\otimes\mathbf{I}^{(i)} + |1\rangle\langle 1|^{(j)}\otimes\mathbf{Z}^{(i)}, \tag{13}$$

where the gate apply $\mathbf{Z}(\mathbf{I})$ on the ith qubit if the jth qubit in the state $|1\rangle(|0\rangle)$.

3.4 Classical Data Encoding

When using VQC for classical data processing, the classical data need to be encoded into a quantum state. Various data encoding methods are available, including basis encoding, amplitude encoding, and angle encoding [30]. The comparative analysis of these encoding methods is summarized in Table 1, and the qubit resource requirement are shown in Fig. 6.

Basis Encoding. Basis encoding is a prevalent information encoding method in qubit-based quantum computing. This method is commonly applied in cases where the inputs consist of binary strings with a length of N and the input space i is defined as $i = \{0, 1\}^{\otimes N}$. To encode an arbitrary vector v in basis encoding, first, we need to translate each value of v into its binary string that follows binary fraction representation

$$v = \sum_{k=0}^{N-1} i_k \frac{1}{2^k}, \tag{14}$$

then these bit strings can be concatenated and represented by the quantum state. Having N qubits to encode each feature, the number of qubits needed to do basis encoding is NM, where M is the number of features.

Amplitude Encoding. Amplitude encoding is the process of encoding a classical data point x, which can be real or complex-valued and has M dimensions, into the amplitudes of a quantum state with n qubits

$$|\psi\rangle = \sum_{i=0}^{M-1} x_i |k\rangle, \tag{15}$$

where $M = 2^n$, x_i is the ith element of x, and $|k\rangle$ is the ith computational basis state. The main benefit of amplitude encoding is that we only need $n = \log M$ qubits to encode an input with M data features and $n = \log(NM)$ qubits if we have N of those inputs. Nevertheless, it is known that the depth of a state preparation circuit employing amplitude embedding is 2^n parallel operations, most of which are expensive 2-qubit gates. [20].

Angle Encoding. In angle encoding, the classical data point x is mapped to its corresponding qubit through a Pauli rotation

$$|\psi_x\rangle = \bigotimes_{i}^{n} \boldsymbol{R}(x_i)|k^n\rangle, \tag{16}$$

where R can be one of Pauli rotation $\boldsymbol{R_X}$, $\boldsymbol{R_Y}$, $\boldsymbol{R_Z}$ and $|k^n\rangle$ is the n qubits initial state that we define (usually $|0\rangle$ state). To implement angle encoding, a total of n qubits are needed, where n represents the number of data points present in the input.

Table 1. Comparative Analysis of Encoding Methods

Encoding Methods	Advantages	Disadvantages
Basis Encoding	Straightforward to implement due to direct one-to-one mapping of data to quantum states	Inefficiency in quantum resource utilization due to its exponential increase with data size
Angle Encoding	Superior quantum resource efficiency compared to basis encoding due to its use of angles to encode information	Increases quantum circuit complexity due to the higher number of gates needed to prepare the desired state vector
Amplitude Encoding	Optimal utilization of quantum resources, as it encodes data in the amplitudes of quantum states	Implementation complexity is high due to increased requirements for gate count and circuit depth, demanding advanced quantum error correction techniques

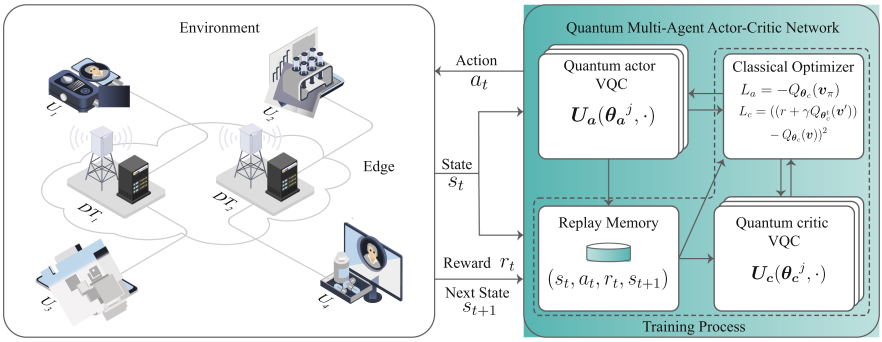

Fig. 5. A Quantum Multi-Agent Actor-Critic Network within a DT Network for Optimal Policy Development. The figure illustrates the integration of quantum multi-agent reinforcement learning in a DT edge network. The network consists of an end layer with entities such as mobile devices and vehicles and an edge layer equipped with MEC servers at base stations. The MEC servers maintain DTs of physical entities. The figure shows the data transfer process, the creation of DTs, and the implementation of VQC for actor and critic networks.

4 Quantum Reinforcement Learning for Digital Twin Placement

4.1 System Model

We consider the following DT edge network, which has the end layer and the edge layer. The end layer comprises entities such as mobile devices and vehicles that have limited computational and storage capabilities. These entities connect to the edge layer via wireless communication and request the available services. The edge layer is made up of base stations that are equipped with edge servers. These servers are utilized to perform computation tasks and provide services to the end-layer entities as depicted in Fig. 5.

In the DT edge network, the MEC servers are responsible for creating and maintaining DTs of the physical entities. The number of physical entities in the end layer is typically much larger than the number of MEC servers in the edge layer. As a result, a single MEC server may be responsible for maintaining multiple DTs of physical entities. This is achieved by creating a mapping between the physical entities and the DTs. The MEC servers use this mapping to keep track of the state and behavior of each physical entity and update the corresponding DT accordingly. This approach allows the network to efficiently use the limited resources available in the edge layer while still providing accurate and up-to-date DTs of the physical entities.

The total time needed for creating a DT of user i at base station j is written as

$$T_{ij} = U_{ij} + C_{ij}, \tag{17}$$

where U_{ij} is the time needed for uploading data of user i, D_i to base station j that is given as,

$$U_{ij} = \frac{D_i}{r_{ij}}, \tag{18}$$

and C_{ij} is the time needed for DT computation that is given as,

$$C_{ij} = \frac{D_i}{c_{ij}}. \tag{19}$$

Here, c_{ij} denotes the computation resource of base station j for DT of user i and r_{ij} is the uploading rate of user i to base station j. The data rate is given by,

$$r_{ij} = B \log \left(1 + \frac{p_i h_{ij}}{\sigma^2}\right), \tag{20}$$

where σ^2 is the noise power, p_i is the transmit power of user i, h_{ij} is the channel gain between user i and base station j, and B is the channel bandwidth.

4.2 Problem Formulation

Let $\boldsymbol{X}$ is a matrix where its element x_{ij} corresponds to the decision variable of assigning the users with the base stations as its DT server, where $x_{ij} = 1$ if the user i use the base station j as its DT server and $x_{ij} = 0$ otherwise. The problem

of DT placement is formulated as

$$\min_{\mathbf{X}} \sum_{i=1}^{N} \sum_{j=1}^{M} T_{ij} x_{ij} \tag{21}$$

$$\text{s.t.} \sum_{j \in M} x_{ij} = 1, \ \forall i \in N \tag{22}$$

$$\sum_{i \in N} x_{ij} \leq M_j, \ \forall j \in M \tag{23}$$

$$\sum_{i \in N} x_{ij} c_{ij} \leq C_j, \ \forall j \in M \tag{24}$$

$$x_{ij} \in \{0, 1\}, \ \forall (i, j) \in (N, M) \tag{25}$$

$$T_{ij} \leq \tau_i. \tag{26}$$

Constraint (22) ensures each user is only served by one base station. Constraint (23) ensures that the MEC server doesn't exceed its maximum number of users M_j. Constraint (24) ensures that the base station does not exceed its computation capacity C_j. Constraint (25) denotes that x_{ij} is a boolean variable. Constraint (26) ensures that the delay requirement τ_i for each user is fulfilled.

4.3 Multi-agent Quantum Reinforcement Learning

Multi-Agent QRL uses VQC for the actor and critic network to extract the relevant features. To encode the classical data into a quantum state, amplitude encoding is used where it needs qubits that scales logarithmicly with the number of input data. The amplitude encoding is given by (14).

For quantum data processing, each layer consist of single parameter rotation on each qubit as,

$$\boldsymbol{R}(\boldsymbol{\gamma}) = \bigotimes_{i} \boldsymbol{R}_X(\gamma_i), \tag{27}$$

which followed by entanglement operation, controlled-Z gates that are applied with cyclic pattern as,

$$\boldsymbol{E} = \prod_{i=N_q}^{1} \boldsymbol{C_X}_{i+1}^{\ i}, \tag{28}$$

where we use $i + 1$ is taken modulo N_q and N_q is the number of qubits.

Given the total number of layer L, the VQC is given by,

$$\boldsymbol{U}(\boldsymbol{\theta}) = \prod_{l=L}^{1} \boldsymbol{E}\boldsymbol{R}(\boldsymbol{\gamma}^l), \tag{29}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\gamma^1}, ..., \boldsymbol{\gamma^L}\}$.

The quantum circuit is measured using observable $\boldsymbol{Z}^{\otimes N_q}$. For classical input $\boldsymbol{y}$ which amplitude encoded to quantum state $|\psi(\boldsymbol{y})\rangle$ the expected value for each qubit is calculated as,

$$e_i = \langle \psi(\boldsymbol{y}) | \boldsymbol{U}^{\dagger}(\boldsymbol{\theta}) \boldsymbol{Z_i} \boldsymbol{U}(\boldsymbol{\theta}) | \psi(\boldsymbol{y}) \rangle, \tag{30}$$

here $\boldsymbol{Z_i} = \bigotimes_k^N \boldsymbol{Z}^c$ where $c = 1$ when $k = i$ and $c = 0$ otherwise. The measurement outcome $\boldsymbol{e}$ is fed into classical machine learning that uses fully connected layers.

We use multi-agent QRL, where each base station is an agent. Each agent observes its assignment variables as state $\boldsymbol{s}^j = \{x_{1j}, ..., x_{Nj}\}$ where the policy is learnt through the agent networks which is parameterized by $\boldsymbol{\theta}_a$. The jth agent's action is composed of the assignment variables between the end users and jth MEC server $\boldsymbol{a}^j = \{x_{1j}, ..., x_{Nj}\}$.

For the exploration of the action space, random noise is added to the output of agent network, $\hat{\boldsymbol{a}}$, making the action as,

$$a_i^j = \text{round}(\hat{a}_i + \epsilon), \tag{31}$$

where ϵ is sampled from $[0, 1]$ with uniform distribution and $\text{round}(x) = 1$ if $x > 0.5$, and 0 otherwise.

Since the reinforcement learning algorithm is designed to maximize the reward function, hence the reward function can be written as,

$$R = -\sum_{i=1}^{N}\sum_{j=1}^{M} T_{ij}x_{ij} - P_1 - P_2 - P_3 - P_4, \tag{32}$$

where

$$P_1 = \sum_i \left(\sum_j x_{ij} - 1 \right)^2, \tag{33}$$

$$P_2 = \begin{cases} \sum_j \left(\sum_i x_{ij} - M_j \right)^2, & \text{if } \sum_i x_{ij} \geq M_j \\ 0, & \text{otherwise} \end{cases}, \tag{34}$$

$$P_3 = \begin{cases} \sum_j \left(\sum_i x_{ij} c_{ij} - C_j \right)^2, & \text{if } \sum_i x_{ij} c_{ij} \geq C_j \\ 0, & \text{otherwise} \end{cases}, \tag{35}$$

$$P_4 = \begin{cases} \sum_{i,j} (x_{ij} T_{ij} - \tau_i)^2, & \text{if } x_{ij} T_{ij} \geq \tau_i \\ 0, & \text{otherwise} \end{cases}. \tag{36}$$

The action-value function is learned via the critic networks using state-action as input $\boldsymbol{v} = \{\boldsymbol{s}^1, ..., \boldsymbol{s}^M, \boldsymbol{a}^1, ..., \boldsymbol{a}^M\}$. The action-value function is approximated

by the critic network which is parameterized by $\boldsymbol{\theta}_c$. The target actor and target critic networks have the same architecture with the actor and critic network respectively. The target actor and target critic network are parameterized by $\boldsymbol{\theta}_a^t$ $\boldsymbol{\theta}_c^t$ respectively.

For a time step t agent j observes state $\boldsymbol{s}_t^j$ and picks action $\boldsymbol{a}_t^j$, according to $\boldsymbol{a}_t^j$ the agent receives the next state $\boldsymbol{s}_{t+1}^j$ and rewards r_t. The agent then stores $(\boldsymbol{s}_t^j, \boldsymbol{a}_t^j, \boldsymbol{s}_{t+1}^j, r_t)$ in the replay memory. In the training phase, the agent samples the data from the replay memory and updates the critic network using the loss function,

$$L_c = ((r + \gamma Q_{\boldsymbol{\theta}_c^t}(\boldsymbol{v}')) - Q_{\boldsymbol{\theta}_c}(\boldsymbol{v}))^2, \tag{37}$$

where Q is the approximated action-value function. The actor is updated with the loss function,

$$L_a = -Q_{\boldsymbol{\theta}_c}(\boldsymbol{v}_\pi), \tag{38}$$

where $\boldsymbol{v}_\pi = \{\boldsymbol{s}^1, ..., \boldsymbol{s}^M, \boldsymbol{a}^1, ..., \boldsymbol{a}^M\}$.

After the training, the target network parameter is updated as,

$$\boldsymbol{\theta}_c^t \leftarrow \boldsymbol{\theta}_c, \tag{39}$$

$$\boldsymbol{\theta}_a^t \leftarrow \boldsymbol{\theta}_a. \tag{40}$$

5 Performance Evaluation

5.1 Simulation Setup

To carry out the simulation for QRL, we utilize Pennylane, an open-source library developed by Xanadu. Pennylane is instrumental in forming a connection between conventional machine learning techniques and the emerging realm of quantum computing. It provides a suite of tools and abstractions that enable the creation and definition of quantum circuits and operators, along with the training of quantum models using traditional data sets. Moreover, Pennylane harmoniously integrates with established TensorFlow modules, enabling the simultaneous execution of both classical and quantum neural networks.

For the simulation, we used TensorFlow v2.7.0, Pennylane v0.31.1. The agent network employed $\log_2(N)$ qubits, while the critic network utilized $\log_2(2MN)$ qubits. The simulation comprised N_e episodes, each consisting of N_t time steps. The optimizer used was Adam with learning rates of 0.001 We set the discount factor for the reward to $\gamma = 0.99$. Lastly, the replay memory size was N_m, and the batch size was N_b. The classical neural network in actor network consist of 3 fully connected layers with relu activation function using 960 learnable parameters for the first layer, 37056 learnable parameters for the second and third layers. The actor has additional fully connected layer with sigmoid activation function with 772 learnable parameters. Meanwhile for the critic networks use 3 fully connected

Algorithm 1. Multi-Agent QRL for DT Placement

Require: Users and Base Stations
Ensure: Policy π for DT placement
  for $j \in 1 \rightarrow M$ **do**
    Initialize replay memory
    Initialize actor network with $\boldsymbol{\theta_a}^j$
    Initialize critic network with $\boldsymbol{\theta_c}^j$
    $\boldsymbol{\theta_a^t}^j \leftarrow \boldsymbol{\theta_a}^j$
    $\boldsymbol{\theta_c^t}^j \leftarrow \boldsymbol{\theta_c}^j$
  end for
  for each episode **do**
    for each time step t **do**
      for $j \in 1 \rightarrow M$ **do**
        Observe the state s_t and choose the action a_t
        Receive reward r_t and observe new state s_{t+1}
        Store (s_t, a_t, r_t, s_{t+1}) in replay memory
      end for
    end for
    for $j \in 1 \rightarrow M$ **do**
      Sample the replay memory
      Train the actor and critic networks
      Update the target networks
    end for
  end for
  Return the policy π

layers with relu activation function using 3264 learnable parameters for the first layer, 37056 learnable parameters for the second and third layers. The actor has additional fully connected layer with no activation function with 193 learnable parameters. Adding quantum layers adding additional learnable parameters to the actor and critic networks. For the actor network, the quantum circuit has additional 2, 6, and 12 learnable parameters for quantum circuit using 1, 3, and 6 layers, respectively. For critic networks the quantum circuit add 4, 12, and 24 learnable parameters for quantum circuit using 1, 3, and 6 layers Table 2.

5.2 Simulation Results and Discussions

We simulate the instance of DT placement problem with 4 users and 2 base stations. The users use 0.1 Watts transmit power with data size range $[300, 500]$ Mb, and the base station have computation power in $[1, 2]$ GHz range. QRL agents are trained for 2000 episodes. For training, a batch size of 128 is used with the Adam optimizer using a learning rate of 0.001 [25].

Figure 7 illustrates the reward achieved by the QRL agents over increasing episodes, with quantum layers varied between 1, 3, and 6, and classical neural network layers fixed at 3. The figure reveals a clear trend: as the number of quantum layers increases, the agents earn better rewards, demonstrating enhanced

Table 2. Simulation Parameters

Parameter	Value
User transmit power p_i	0.1 W
Channel bandwidth B	10 MHz
Noise power σ^2	–174 dBm
Server computation capacity C_j	$[1, 2]$ GHz
Number of User N	4
Number of Base Station M	2
Data size	$[300, 500]$ Mb
Reward discount factor γ	0.99
Training episode	2000
Batch size	128
Classical Optimizer	Adam
Learning rate	0.001

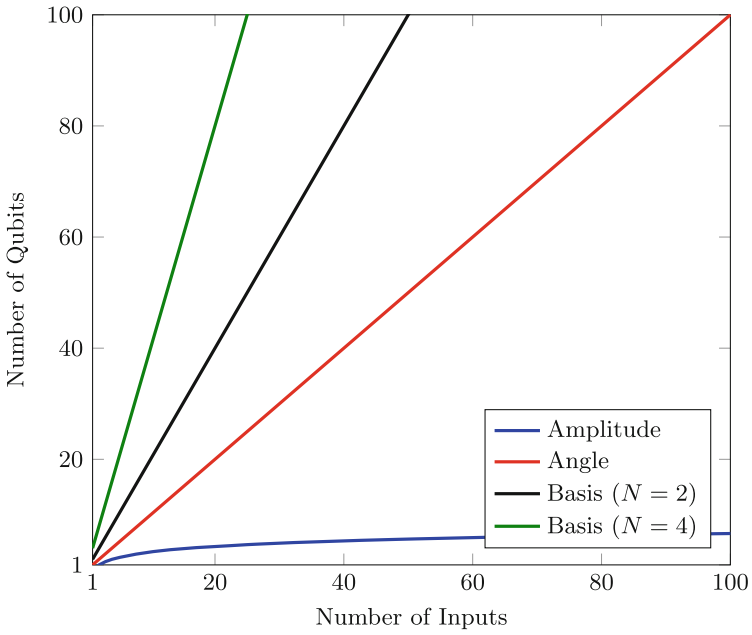

Fig. 6. The number of qubits needed for encoding the classical data via different encoding methods. In the basis encoding methods, the N represents the number of qubits needed for encoding an element of the input data.

robustness in policy learning. Specifically, using 6 quantum layers results in enhanced learning, as the increased complexity allows for more VQCs to optimize the policy. This complexity enables the VQCs to extract valuable features

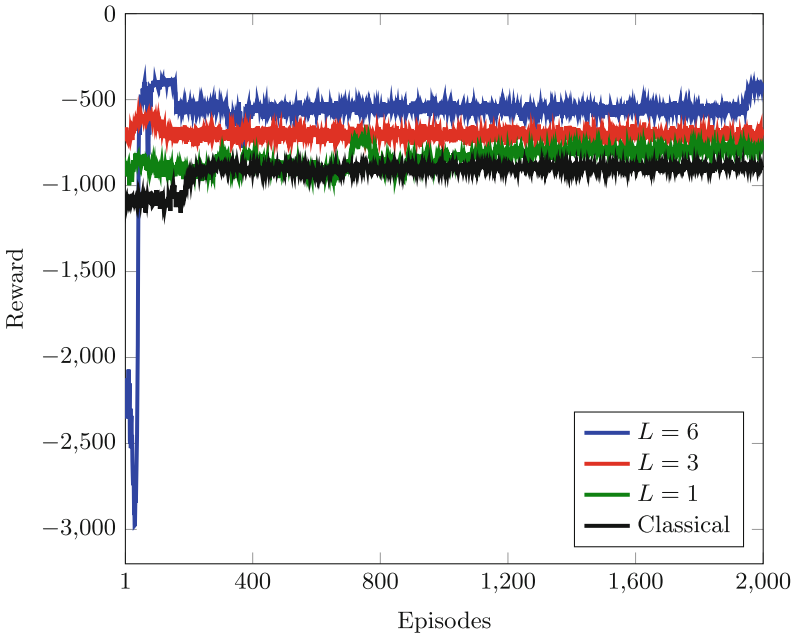

Fig. 7. The reward is achieved by quantum multi-agent reinforcement learning as the number of episodes grows. The reward is plotted using a different number of layers in the VQC of each agent.

and refine the reward function, outperforming the classical neural network, which consists of only 4 fully connected layers for both actor and critic networks. The comparison underscores the potential advantages of incorporating quantum layers in reinforcement learning, particularly in scenarios that demand nuanced policy optimization.

Table 3 shows the latency of the users for both classical and quantum layers. The classical approach yields an average latency of 1.756590952, while the quantum layers demonstrate a more optimized performance. Agents with a single-layer quantum network (Layer 1) still show comparable performance, with an average latency of 1.655676468. The use of two extra quantum layers (layers 3 and 6) indicates the presence of a learning process in which the agents explore potential solutions. Specifically, agents with 3 layers have an average latency of 1.492901308, while those with 6 layers have an average latency of 1.26179161. This trend suggests that agents with more layers in their quantum networks can learn a better policy to minimize the latency of the network while trying to fulfill the constraints by optimizing the reward function. The bold values in the table highlight the minimum latency for each user, showcasing that the quantum approach often results in lower latency compared to the classical method. The comparison between classical and quantum layers emphasizes the advantage of quantum layers in policy learning. The decreasing average latency across the

Table 3. Latency comparison of different users

User	Classical	Quantum Layer		
		1	3	6
User 1	1.998166767	1.965989006	1.648570744	**1.094460226**
User 2	1.79182047	1.534144303	1.360584811	**1.116585376**
User 3	1.939092125	**1.36118947**	1.515029339	1.776926329
User 4	1.297284444	1.761383092	1.447420337	**1.059194508**
Average	1.756590952	1.655676468	1.492901308	**1.26179161**

quantum layers, contrasted with the classical layer, demonstrates the system's ability to optimize and reduce latency as the complexity of the quantum network increases. This evidence supports the conclusion that using quantum layers results in better policy learning compared to the classical approach. As we have 4 users, the table also shows the individual latencies for each user across the different layers.

6 Conclusion

In this paper, we have studied the application of multi-agent QRL to enhance the efficiency of DT deployment in the complex multi-tier framework of 6G networks. Our investigation highlights the robustness of multi-agent QRL in terms of policy learning, as well as its distinctive resource efficiency with regard to learnable parameters per layer in comparison to classical counterparts. This suggests that the multi-agent QRL can serve as a lightweight, pragmatic, and high-performance solution for MEC. Since we are currently navigating through the NISQ era where quantum computers are not yet perfect and suffer from noise-related issues. However, substantial efforts are being made from academic and industrial domains to address these challenges, enabling a transition towards a post-NISQ era. Intriguingly, even within the constraints of the NISQ era, our study demonstrates that VQC-based algorithms can still exhibit competitive performance, even on noisy quantum computers, holding their own against classical algorithms. This study underscores the potential of quantum-centric methodologies in enhancing network performance and lays the groundwork for potential advancements in 6G network optimization algorithms.

References

1. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: a survey. IEEE Internet Things J. **5**(1), 450–465 (2018)
2. Ahmed, E., Rehmani, M.H.: Mobile edge computing: Opportunities, solutions, and challenges. Futur. Gener. Comput. Syst. **70**, 59–63 (2017)
3. Anschuetz, E.R., Hu, H.Y., Huang, J.L., Gao, X.: Interpretable quantum advantage in neural sequence learning. PRX Quant. **4**(2), 020338 (2023)

4. Azad, U., Behera, B.K., Ahmed, E.A., Panigrahi, P.K., Farouk, A.: Solving vehicle routing problem using quantum approximate optimization algorithm. IEEE Trans. Intell. Transp. Syst. (2022). https://doi.org/10.1109/TITS.2022.3172241, Early access, May 13, 2022, https://doi.org/10.1109/TITS.2022.3172241
5. Botsinis, P., et al.: Quantum search algorithms for wireless communications. IEEE Commun. Surv. Tutor. **21**(2), 1209–1242 (2019)
6. Cerezo, M., et al.: Variational quantum algorithms. Nat. Rev. Phys. **3**(9), 625–644 (2021)
7. Chowdhury, M.Z., Shahjalal, M., Ahmed, S., Jang, Y.M.: 6G wireless communication systems: applications, requirements, technologies, challenges, and research directions. IEEE Open J. Commun. Soc. **1**, 957–975 (2020). https://doi.org/10.1109/OJCOMS.2020.3010270
8. Chukhno, O., Chukhno, N., Araniti, G., Campolo, C., Iera, A., Molinaro, A.: Placement of social digital twins at the edge for beyond 5G IoT networks. IEEE Internet Things J. **9**(23), 23927–23940 (2022)
9. Duong, T.Q., Van Huynh, D., Khosravirad, S.R., Sharma, V., Dobre, O.A., Shin, H.: From digital twin to metaverse: the role of 6g ultra-reliable and low-latency communications with multi-tier computing. IEEE Wireless Commun. **30**(3), 140–146 (2023)
10. Ghildiyal, Y., et al.: An imperative role of 6g communication with perspective of industry 4.0: challenges and research directions. Sustain. Energy Technol. Assess. **56**, 103047 (2023)
11. Havlíček, V., et al.: Supervised learning with quantum-enhanced feature spaces. Nature **567**(7747), 209–212 (2019). https://doi.org/10.1038/s41586-019-0980-2
12. Henderson, M., Gallina, J., Brett, M.: Methods for accelerating geospatial data processing using quantum computers. Quantum Mach. Intell. **3**(1) (2021)
13. Jerbi, S., Trenkwalder, L.M., Nautrup, H.P., Briegel, H.J., Dunjko, V.: Quantum enhancements for deep reinforcement learning in large spaces. PRX Quant. **2**(1) (2021)
14. Khan, L.U., Han, Z., Saad, W., Hossain, E., Guizani, M., Hong, C.S.: Digital twin of wireless systems: overview, taxonomy, challenges, and opportunities. IEEE Commun. Surv. Tutorials **24**(4), 2230–2254 (2022)
15. Khan, L.U., Saad, W., Niyato, D., Han, Z., Hong, C.S.: Digital-twin-enabled 6G: Vision, architectural trends, and future directions. IEEE Commun. Mag. **60**(1), 74–80 (2022)
16. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
17. Lu, Y., Zheng, X.: 6g: a survey on technologies, scenarios, challenges, and the related issues. J. Ind. Inf. Integr. **19**, 100158 (2020)
18. Lu, Y., Maharjan, S., Zhang, Y.: Adaptive edge association for wireless digital twin networks in 6G. IEEE Internet Things J. **8**(22), 16219–16230 (2021). https://doi.org/10.1109/JIOT.2021.3098508
19. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
20. Möttönen, M., Vartiainen, J.J., Bergholm, V., Salomaa, M.M.: Quantum circuits for general multiqubit gates. Phys. Rev. Lett. **93**, 130502 (2004). https://doi.org/10.1103/PhysRevLett.93.130502
21. Nang Paing, S., Setiawan, J.W., Tariq, S., Talha Rahim, M., Lee, K., Shin, H.: Counterfactual anonymous quantum teleportation in the presence of adversarial attacks and channel noise. Sensors **22**(19), 7587 (2022)

22. Neill, C., Roushan, P., Kechedzhi, K., et al.: A blueprint for demonstrating quantum supremacy with superconducting qubits. Science **360**(6385), 195–199 (2018). https://doi.org/10.1126/science.aao4309
23. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information: 10th Anniversary Edition, Cambridge University Press (2010)
24. Park, C., et al.: Quantum multi-agent actor-critic networks for cooperative mobile access in multi-UAV systems. IEEE Internet Things J. 1–1 (2023)
25. Park, C., .: Quantum multi-agent actor-critic networks for cooperative mobile access in multi-UAV systems. IEEE Internet Things J. (2023)
26. Park, J., et al.: Extreme ultra-reliable and low-latency communication. Nat. Electron. **5**(3), 133–141 (2022)
27. Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)
28. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. Phys. Rev. Lett. **113**(13), 130503 (2014)
29. Saggio, V., et al.: Experimental quantum speed-up in reinforcement learning agents. Nature **591**(7849), 229–233 (2021)
30. Schuld, M.: Supervised quantum machine learning models are kernel methods. arXiv:2101.11020 (2021)
31. Sutton, R.S., Barto, A.G.: Reinforcement learning: an introduction, MIT press (2018)
32. Vaezi, M., Noroozi, K., Todd, T.D., Zhao, D., Karakostas, G.: Digital twin placement for minimum application request delay with data age targets. IEEE Internet Things J. **10**(13), 11547–11557 (2023)
33. Wang, C., Rahman, A.: Quantum-enabled 6G wireless networks: opportunities and challenges. IEEE Wireless Commun. **29**(1), 58–69 (2022). https://doi.org/10.1109/MWC.006.00340
34. Wang, D., Song, B., Lin, P., Yu, F.R., Du, X., Guizani, M.: Resource management for edge intelligence (EI)-assisted IoV using quantum-inspired reinforcement learning. IEEE Internet Things J. **9**(14), 12588–12600 (2022)
35. Wang, Z., Hadfield, S., Jiang, Z., Rieffel, E.G.: Quantum approximate optimization algorithm for MaxCut: a fermionic view. Phys. Rev. A **97**(2), 022304 (2018). https://doi.org/10.1103/physreva.97.022304
36. Wu, Y., Zhang, K., Zhang, Y.: Digital twin networks: a survey. IEEE Internet Things J. **8**(18), 13789–13804 (2021)