

Variational Quantum Rainbow Deep Q-Network for Optimizing Resource Allocation Problem

Truong Thanh Hung Nguyen
Analytics Everywhere Lab,
University of New Brunswick
Fredericton, Canada
hung.ntt@unb.ca

Truong Thinh Nguyen
University of Science and Technology
of Hanoi
Hanoi, Vietnam
thinht2410944@usth.edu.vn

Hung Cao
Analytics Everywhere Lab,
University of New Brunswick
Fredericton, Canada
hcao3@unb.ca

Abstract

Resource allocation remains NP-hard due to combinatorial complexity. While deep reinforcement learning (DRL) methods, such as the Rainbow Deep Q-Network (DQN), improve scalability through prioritized replay and distributional heads, classical function approximators limit their representational power. We introduce *Variational Quantum Rainbow DQN* (VQR-DQN), which integrates ring-topology variational quantum circuits with Rainbow DQN to leverage quantum superposition and entanglement. We frame the human resource allocation problem (HRAP) as a Markov decision process (MDP) with combinatorial action spaces based on officer capabilities, event schedules, and transition times. On four HRAP benchmarks, VQR-DQN achieves 26.8% normalized makespan reduction versus random baselines and outperforms Double DQN and classical Rainbow DQN by 4.9-13.4%. These gains align with theoretical connections between circuit expressibility, entanglement, and policy quality, demonstrating the potential of quantum-enhanced DRL for large-scale resource allocation.

CCS Concepts

• **Hardware** → Quantum computation; • **Theory of computation** → Reinforcement learning.

Keywords

Quantum reinforcement learning, resource allocation problem

ACM Reference Format:

Truong Thanh Hung Nguyen, Truong Thinh Nguyen, and Hung Cao. 2026. Variational Quantum Rainbow Deep Q-Network for Optimizing Resource Allocation Problem. In *Proceedings of The 41st ACM/SIGAPP Symposium on Applied Computing (SAC'26)*. ACM, New York, NY, USA, 8 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 Introduction

Resource allocation represents a fundamental NP-hard combinatorial optimization problem with diverse applications across software systems, including underwater resource management [43, 4], human resource allocation [27, 28], inventory allocation [36,

37], and network resource distribution [19, 2]. Various solution approaches have been developed due to the problem's complexity [29, 13]. While exact classical methods succeed for small problem instances, real-world software engineering scenarios often involve high-dimensional optimization spaces where classical approaches encounter exponential time complexity, creating a critical performance bottleneck. This computational barrier has positioned quantum computing as a promising paradigm for addressing resource allocation challenges in modern software systems, leveraging quantum mechanical principles to explore solution spaces more efficiently than classical counterparts.

Reinforcement Learning (RL) has emerged as a promising approach for addressing these challenges. Deep Q-Networks (DQN) utilize neural networks to map state spaces to action Q-values, enabling optimal action selection [25]. The RL field has demonstrated impressive capabilities across games [32], continuous control [21], locomotion [30], navigation [8], and robotics [20], while showing effectiveness in solving optimization problems relevant to resource allocation [6]. Recent quantum computing advancements have introduced new possibilities, particularly through Quantum Reinforcement Learning (QRL). Quantum computing leverages superposition and entanglement to explore solution spaces infeasible for classical computers [35]. A key QRL approach utilizes Variational/Parameterized Quantum Circuits (VQCs/PQCs) [18, 34], optimizable with classical ML techniques. VQCs function as quantum feature extractors, capturing complex data correlations challenging for classical models, thereby enhancing RL agent representation capabilities for improved decision-making in complex environments.

Hence, we introduce a QRL framework called the *Variational Quantum Rainbow Deep Q-Network* (VQR-DQN), integrating VQC-based quantum feature extraction with advanced RL techniques. Our key contributions are: **(1) VQR-DQN Framework:** A novel architecture combining quantum-enhanced Ring-topology VQCs with Rainbow DQN [15], incorporating distributional Q-learning, prioritized replay, and noisy networks. **(2) Resource Allocation Environment:** We demonstrate the VQR-DQN effectiveness via the Human Resource Allocation Problem (HRAP) as an MDP with a comprehensive environment design simulating real-world personnel dispatch scenarios. **(3) Experimental Evaluation:** Extensive experiments demonstrating VQR-DQN's significant outperformance over Double DQN [38] and Rainbow DQN [15] in task completion time and resource utilization across varying HRAP complexity scenarios.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SAC'26, Thessaloniki, Greece

© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-X-XXXX-XXXX-X/26/03
<https://doi.org/XXXXXXX.XXXXXXX>

2 Related Work

2.1 Resource Allocation Problem

Resource allocation problem represents a long-standing problem in operations research and management science. Traditional approaches employ mathematical optimization techniques, i.e., Linear Programming (LP) [5], Mixed-Integer Linear Programming (MILP) [14], and branch-and-bound algorithms (B&B) [39], modeling allocation problems as linear equations with constraints. However, these methods are limited to small-scale problems due to exponential computational complexity growth. Heuristic and metaheuristic approaches, such as Genetic Algorithms (GA) [26], Simulated Annealing (SA) [22], and Particle Swarm Optimization (PSO) [17], provide approximate solutions for large-scale problems [7], yet suffer from domain-specific parameter tuning requirements and limited adaptability. RL has emerged as a promising resource allocation problem paradigm. Early applications used tabular approaches like Q-learning [41] for small state-action spaces. Deep RL (DRL) advancement enabled high-dimensional problem handling through DQN and variants, i.e., DDQN [38] and Dueling DQN [40], applied to this problem using neural networks for action-value function approximation [29, 13]. Recent work [27] has integrated RL with search-based methods like Monte Carlo tree search (MCTS) [32], combining RL and heuristic search strengths for complex dynamic environments. Challenges remain in scaling RL methods to real-world resource allocation problems, particularly regarding high-dimensional state representations, policy robustness, and convergence efficiency.

2.2 Quantum Reinforcement Learning (QRL)

QRL emergence has introduced new opportunities for complex optimization problems like HRAP. Primary advantages include handling exponentially large state-action spaces and simultaneous exploration of multiple solutions via quantum superposition and entanglement [24, 11, 9]. VQCs enhance agent environment understanding by extracting high-dimensional features difficult for classical methods. Recent research explores VQCs [18, 34, 1] as quantum feature extractors within RL frameworks, encoding classical data into quantum states and applying parameterized quantum gates to capture complex data correlations. While QRL application to resource allocation tasks remains emerging [3, 1, 42], HRAP applications show significant promise. Integrating quantum computing techniques into existing RL frameworks aims to overcome classical approach limitations and achieve improved scalability and performance in real-world resource allocation tasks.

3 Environment

Our HRAP environment, as illustrated in Figure 1, is formulated as an MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space representing the environment at each time step, $\mathcal{A}$ is the action space, representing task assignments for officers, $P(s'|s, a)$ is the transition probability, which defines how the environment evolves based on the agent's actions, $R(s, a)$ is the reward function, representing the feedback signal for the agent's actions, γ is the discount factor, which prioritizes immediate rewards over future rewards.

In RL, the agent interacts with the environment by observing the current state, selecting an action, and receiving feedback in the form of a transition tuple:

$$(s_t, a_t, r_t, s_{t+1}, d_t), \quad (1)$$

where s_t is the current state at time step t , a_t is the action taken by the agent in state s_t , r_t is the reward received from the environment after taking action a_t , s_{t+1} is the next state resulting from the action a_t , d_t is the termination flag indicates whether the episode has terminated. At each time step t , the agent observes s_t , selects an action a_t from the available action space, and receives a reward r_t . The agent then transitions to a new state s_{t+1} and repeats the process until the episode ends, as indicated by $d_t = 1$. The agent aims to learn an optimal policy that maximizes the cumulative reward over time by accurately estimating the expected future rewards, or Q -values.

Entities. Our HRAP environment consists of three primary entities: Officers, Events, and Tasks, which define the core elements of the problem: **(1) Officers:** Let O denote the number of officers, where each officer $o \in \{1, \dots, O\}$ is assigned to perform specific tasks across various events. **(2) Events:** The set of scheduled events, each consisting of multiple tasks. Let E denote the number of events, indexed by $e \in \{1, \dots, E\}$, where each event requires the completion of all associated tasks within a specified timeframe. **(3) Tasks:** The set of tasks within each event. Let T denote the number of tasks per event, indexed by $t \in \{1, \dots, T\}$.

Objective. The objective is to assign officers to tasks in a manner that minimizes the maximum completion time across all events. The completion time for an event is determined by the time taken to complete all its tasks, considering both task execution times and transition times between events.

3.1 State Space

The state space $\mathcal{S}$ represents the current configuration of the environment, containing all the necessary information for the agent to make informed decisions. In the HRAP environment, the state s_t is a high-dimensional vector comprising the following components:

Officers' Capability Matrices $C_o \in \mathbb{Z}^{E \times T}$ for each officer o . Each entry $C_{o,e,t}$ represents the time required for officer o to complete task t in event e . The capability matrices are initialized with random integer values uniformly sampled from the interval $[1, 20]$.

Event Occurrence Times $\Omega \in \mathbb{Z}^E$, where Ω_e denotes the start time of event e . These occurrence times are randomly generated as integers from the interval $[1, 20]$ and are sorted in ascending order to ensure temporal ordering.

Transition Matrix $\mathbf{M} \in \mathbb{Z}^{(E+1) \times (E+1)}$, representing the time required for an officer to travel between events. Each entry M_{e_1, e_2} indicates the transition time from event e_1 to event e_2 . The matrix is symmetric with zero diagonal entries, ensuring no time is required to remain at the same event.

The state of the environment is represented by the concatenation of the flattened capability matrices of all officers' capabilities, event occurrence times, and the transition matrix:

$$s = \text{Concat}(\text{Flatten}(C_1, \dots, C_N), \Omega, \text{Flatten}(\mathbf{M})) \in \mathbb{R}^d, \quad (2)$$

$$d = O \times E \times T + E + (E + 1)^2.$$

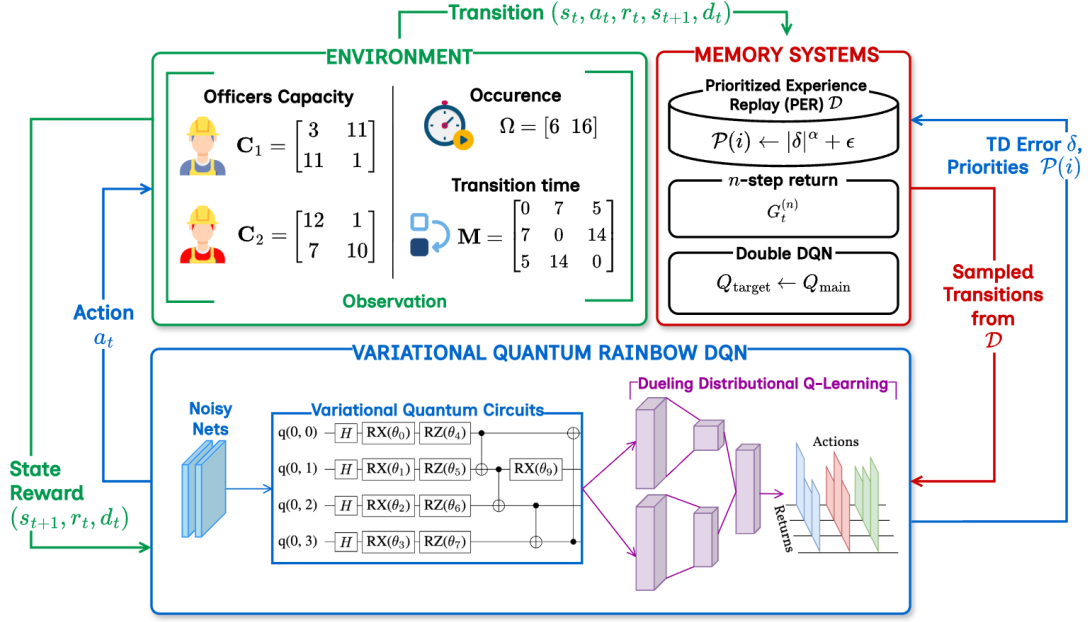


Figure 1: Architecture of VQR-DQN in solving HRAP environment. Ring-topology VQCs are integrated into the Rainbow DQN pipeline, combining noisy exploration, prioritized replay, n -step returns, Double DQN, and dueling distributional Q -learning.

3.2 Action Space

The action space $\mathcal{A}$ defines the possible task assignments the agent can make at each time step t . An action a_t consists of assigning an officer to a task within an event. For each task t in event e , the agent selects an officer o from the pool of available officers:

$$a_t = \{(e, t, o) \mid e \in \{1, \dots, E\}, t \in \{1, \dots, T\}, o \in \{1, \dots, O\}\}, \quad (3)$$

given E events, each with T tasks, and O officers, the total number of possible actions is $O^{E \times T}$, making the action space combinatorially large.

3.3 Reward Function

The reward function $R(s, a)$ is designed to motivate efficient task allocations by minimizing the maximum time taken to complete any event. At each time step, the agent receives a reward r_t based on the negative completion time for the slowest event, normalized by the maximum possible completion time Ψ . The maximum completion time Ψ is defined as:

$$\Psi = (\max(\mathbf{C}) \times E \times T) + (\max(\mathbf{M}) \times E \times T). \quad (4)$$

The reward r_t is then calculated as:

$$r_t = -\frac{\max_e (\sum_t C_{o,e,t} + \sum_{\text{transitions}} M_{e_1,e_2})}{\Psi}. \quad (5)$$

This reward structure encourages the agent to minimize the longest completion time across all events, promoting efficient task assignments and travel schedules. By normalizing the reward with Ψ , the reward values are scaled to a consistent range, improving the learning's stability and convergence.

4 Variational Quantum Rainbow Deep Q-Network (VQR-DQN)

We propose the *Variational Quantum Rainbow Deep Q-Network* (VQR-DQN), a DRL framework integrating VQCs as quantum-enhanced feature extractors with Rainbow DQN mechanisms (dueling distributional Q -learning, noisy exploration, prioritized replay, n -step returns, and DDQN). Leveraging quantum-enhanced feature extraction with Ring topology, VQR-DQN captures complex correlations within high-dimensional, entangled HRAP state spaces comprising officers' capabilities, event occurrences, and transition matrices.

4.1 Variational Quantum Circuits (VQCs)

In our VQR-DQN framework, VQCs serve as high-dimensional, entangled feature extractors employing parameterized quantum operations in Ring topology for quantum data transformations. Detailed definitions of fundamental VQCs' terminology are provided in Appendix A.

4.1.1 Circuit Architecture. Our designed VQCs comprise multiple layers, each consisting of parameterized single-qubit rotations followed by entangling CNOT gates arranged in a Ring topology. Figure 2 visualizes our ansatz with 4 input qubits and 2 layers. This ansatz architecture ensures global entanglement across all qubits, boosting the capture of complex feature correlations.

Let n_q denote the number of qubits and n_l the number of layers in the VQC. The quantum state evolves through each layer $l \in \{1, \dots, n_l\}$ as follows:

Initialization. Each qubit is initialized in the ground state $|0\rangle$, and a layer of Hadamard gates H is applied to create an equal

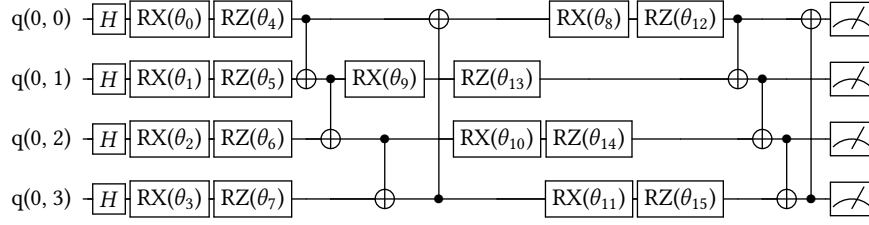


Figure 2: The visualization of the ansatz with the Ring topology with 4 input qubits and 2 layers in our designed VQC.

superposition:

$$|\psi_0\rangle = \bigotimes_{i=1}^{n_q} |0\rangle_i, \quad |\psi_{\text{init}}\rangle = H^{\otimes n_q} |\psi_0\rangle, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \quad (6)$$

Parameterized Rotations. For each qubit i in layer l , we apply parameterized rotation gates $RX(\theta_{l,i}^{(x)})$ and $RZ(\theta_{l,i}^{(z)})$ for Pauli-X and Pauli-Z, respectively:

$$\begin{aligned} RX(\theta) &= e^{-i\theta X/2} = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}, \\ RZ(\phi) &= e^{-i\phi Z/2} = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix}. \end{aligned} \quad (7)$$

Entangling Gates. Following the rotations, we introduce entanglement using Controlled-NOT (CNOT) gates in a Ring topology. The CNOT gate between qubits q_i (control) and q_{i+1} (target) is represented by the matrix:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \quad (8)$$

For n_q qubits, the entangling layer applies CNOT gates as follows: $\text{CNOT}(q_i \rightarrow q_{(i+1) \bmod n_q})$ for each qubit $i \in \{1, \dots, n_q\}$. In total, each layer l produces a unitary $U^{(l)}(\theta)$ comprising all single-qubit rotations and entangling gates.

Measurement and Final Quantum State. For each layer l , the composite unitary operation applied to all qubits is:

$$U^{(l)}(\theta^{(l)}) = \left(\bigotimes_{i=1}^{n_q} RX(\theta_{l,i}^{(x)}) \cdot RZ(\theta_{l,i}^{(z)}) \right) \cdot \text{CNOT}_{\text{Ring}}. \quad (9)$$

After n_l layers, the cumulative unitary $U(\theta)$ is: $U(\theta) = \prod_{l=1}^{n_l} U^{(l)}(\theta^{(l)})$. Thus, the final quantum state is: $|\psi_{\text{out}}(\theta)\rangle = U(\theta) |\psi_{\text{init}}\rangle$. We measure Pauli-Z operators on each qubit: $\hat{Z}_i = I \otimes \dots \otimes Z \otimes \dots \otimes I$, yielding expectation values $\langle \psi_{\text{out}} | \hat{Z}_i | \psi_{\text{out}} \rangle \in [-1, 1]$. Collecting them for $i = 1, \dots, n_q$ yields a quantum feature vector:

$$\mathbf{q}(\theta) = (\langle \hat{Z}_1 \rangle, \dots, \langle \hat{Z}_{n_q} \rangle)^T \in \mathbb{R}^{n_q}, \quad (10)$$

which we feed into subsequent classic layers in the network.

4.2 Rainbow DQN Integration

Our VQR-DQN incorporates all components of Rainbow DQN, augmented by the quantum feature extractor. Below, we detail each component and its role in the overall architecture.

4.2.1 Noisy Networks for Exploration. The VQR-DQN employs noisy dense layers to facilitate exploration by introducing trainable noise into the network parameters. This approach allows the agent to explore without relying solely on the ϵ -greedy policy. The noisy layers modify the weights and biases as follows:

$$\mathbf{w} = \mathbf{w}_\mu + \mathbf{w}_\sigma \odot \boldsymbol{\epsilon}_w, \quad \mathbf{b} = \mathbf{b}_\mu + \mathbf{b}_\sigma \odot \boldsymbol{\epsilon}_b, \quad (11)$$

where $\mathbf{w}_\mu, \mathbf{w}_\sigma$ and $\mathbf{b}_\mu, \mathbf{b}_\sigma$ are trainable parameters, and $\boldsymbol{\epsilon}_w, \boldsymbol{\epsilon}_b$ are noise variables sampled from a Gaussian distribution. The noise injection encourages the network to explore more diverse actions by perturbing the Q -values during training.

4.2.2 Prioritized Experience Replay. The Prioritized Experience Replay (PER) buffer enables the agent to focus more on transitions that have a higher learning potential by sampling based on the temporal-difference (TD) error δ_i . The sampling probability for each transition i is given by:

$$p_i = \frac{|\delta_i|^\alpha + \epsilon}{\sum_j |\delta_j|^\alpha + \epsilon}, \quad (12)$$

where δ_i is the TD error for the transition is computed as: $\delta_i = r_t + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a') - Q_{\text{main}}(s_t, a_t)$, where α controls the degree of prioritization, and ϵ ensures that all transitions have a non-zero probability of being sampled.

4.2.3 n -step Returns. To provide richer learning signals, the VQR-DQN computes n -step returns for sampled transitions. The n -step return for a transition starting at time step t is defined as:

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n Q_{\text{target}}(s_{t+n}, a_{t+n}), \quad (13)$$

where γ is the discount factor, r_{t+k} is the reward at step $t+k$, and Q_{target} is the target network. The n -step return combines immediate and future rewards over a longer horizon, improving the stability and efficiency of learning.

4.2.4 Double DQN (DDQN). To mitigate overestimation bias, VQR-DQN employs the DDQN technique [38]. The target Q -value is computed using the main network to select actions and the target network to evaluate them:

$$a^* = \arg \max_{a'} Q_{\text{main}}(s', a'; \theta), \quad y = r + \gamma Q_{\text{target}}(s', a^*; \theta^-), \quad (14)$$

where Q_{main} and Q_{target} denote the main and target networks, respectively.

4.2.5 Dueling Distributional Q-Learning. The VQR-DQN integrates the dueling architecture into the distributional Q -learning (C51) framework to improve stability and learning efficiency. In this approach, the Q -value function is decomposed into two streams: **Value Stream** $V(s)$ represents the state value, independent of actions, **Advantage Stream** $A(s, a)$ represents the relative benefit of taking action a in state s . The network outputs two sets of logits, $\mathbf{h}_V(s)$ and $\mathbf{h}_A(s)$, which are transformed into probability distributions using the Softmax function:

$$\begin{aligned} V(s) &= \text{Softmax}(\mathbf{h}_V(s)) \in \mathbb{R}^{1 \times N_{\text{atoms}}}, \\ A(s) &= \text{Softmax}(\mathbf{h}_A(s)) \in \mathbb{R}^{|\mathcal{A}| \times N_{\text{atoms}}} \end{aligned} \quad (15)$$

The final Q -value distribution $\mathbf{p}(s, a)$ for each action is computed by combining the value and advantage streams:

$$\mathbf{p}(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a') \right). \quad (16)$$

This formulation ensures the advantage function has zero mean across actions, making the network more stable. The output $\mathbf{p}(s, a)$ represents a categorical distribution over discrete support points (atoms), capturing the uncertainty in future rewards. The complete VQR-DQN framework is detailed in Algorithm 1, which describes the quantum-enhanced forward pass with Ring-topology VQC feature extraction and dueling distributional heads.

Algorithm 1: VQR-DQN Algorithm

Input: State vector $\mathbf{s} \in \mathbb{R}^{n_{\text{state}}}$

Output: Distribution of Q -values $\mathbf{p}(s, a)$ for each action $a \in \mathcal{A}$

Step 1: Initialize Network Parameters

- Number of qubits n_q , layers in VQC n_l
- Noisy layers with random weights $\mathbf{w}_\mu, \mathbf{w}_\sigma$
- Distributional atoms $\mathcal{Z} = \{z_1, \dots, z_{N_{\text{atoms}}}\}$

Step 2: Input Processing

- $\mathbf{x} \leftarrow \text{NoisyDense}(\mathbf{s}, 512, \text{ReLU})$
- $\mathbf{x} \leftarrow \text{NoisyDense}(\mathbf{x}, 512, \text{ReLU})$

Step 3: Quantum Feature Extraction using VQC

- $\mathbf{q}_{\text{encoded}} \leftarrow \text{Dense}(2 \times n_q \times n_l, \tanh)(\mathbf{x})$
- for** $l = 1$ **to** n_l **do**
 - Apply Hadamard gates to all qubits: $H^{\otimes n_q}$
 - Apply parameterized rotations $\text{RX}(\theta_{l,i}^{(x)})$ and $\text{RZ}(\theta_{l,i}^{(z)})$ to each qubit i
 - Apply Ring entanglement: $\text{CNOT}(q_i \rightarrow q_{(i+1) \bmod n_q})$
- Measure Pauli-Z expectation values: $\mathbf{q}(\theta) \leftarrow [\langle \hat{Z}_1 \rangle, \dots, \langle \hat{Z}_{n_q} \rangle]$

Step 4: Dueling Distributional Head

- Value stream: $V(s) \leftarrow \text{Softmax}(\mathbf{h}_V(s))$
- Advantage stream: $A(s, a) \leftarrow \text{Softmax}(\mathbf{h}_A(s, a))$
- Final Q -value distribution: $\mathbf{p}(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a') \right)$

return Distribution $\mathbf{p}(s, a) \forall a \in \mathcal{A}$

5 Implementation

This section describes the training procedure of the VQR-DQN framework, including network initialization, agent-environment interaction, and network optimization. Algorithm 2 outlines the training procedure incorporating prioritized replay, n -step returns, and target network updates.

5.1 Network Initialization

The VQR-DQN agent consists of a main network Q_{main} and a target network Q_{target} , both initialized with the same parameters at the start of training. The network comprises noisy dense layers followed by a VQC for feature extraction and a dueling distributional head to estimate the Q -value distribution for each action. The target network is updated periodically to stabilize the learning process. The replay buffer $\mathcal{D}$ is initialized as a PER buffer, which allows the agent to sample transitions based on their TD errors. The exploration strategy begins with a high exploration rate ϵ and gradually decays over time using an ϵ -greedy policy.

5.2 Agent-Environment Interaction

Action Selection. Each episode resets environment to initial state s_0 . At time step t , agents select action a_t using ϵ -greedy policy: with probability ϵ , random action from $\mathcal{A}$; otherwise, action maximizing Q -value from main network $Q_{\text{main}}(s_t, a)$. Transition tuples $(s_t, a_t, r_t, s_{t+1}, d_t)$ store in replay buffer, where d_t indicates episode termination.

Exploration-Exploitation Strategy. Exploration rate ϵ follows exponential decay: $\epsilon \leftarrow \max(\epsilon \times \epsilon_{\text{decay}}, \epsilon_{\text{min}})$, where ϵ_{decay} controls decay rate and ϵ_{min} sets lower bound.

5.3 Network Optimization

During each training step, a mini-batch $\mathcal{B}$ of transitions is sampled from the replay buffer $\mathcal{D}$, with sampling probability proportional to the priorities (Eq. 12). For each sampled mini-batch, the agent performs a gradient descent step to minimize the loss between the predicted Q -value distribution and the target distribution. Here, the n -step return $G_t^{(n)}$ is used to compute a more stable target:

$$G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n Q_{\text{target}}(s_{t+n}, a^*), \quad (17)$$

where $a^* = \arg \max_a Q_{\text{main}}(s_{t+n}, a)$. This n -step return provides richer learning signals by incorporating future rewards over a longer horizon.

The loss function is based on the cross-entropy between the predicted distribution and the projected target distribution: $\mathcal{L} = \mathbb{E}_{\mathcal{B}} [\delta^2]$, where δ is the TD error for each transition in the mini-batch. The loss ensures that the Q -value estimates from the main network Q_{main} converge toward the n -step return target. Then, the priorities in the replay buffer are updated based on the new TD errors: $\mathcal{P}(i) \leftarrow |\delta_i|^\alpha + \epsilon$, where α controls the prioritization degree, and ϵ ensures that all transitions have non-zero priority. Gradients are clipped to prevent exploding gradients. The parameters of the main network θ are then updated using the Adam optimizer with a learning rate η . Finally, the target network Q_{target} is periodically synchronized with the main network Q_{main} to stabilize the training.

Algorithm 2: VQR-DQN Initialization and Training Procedure

Input: Replay buffer $\mathcal{D}$, target network Q_{target} , main network Q_{main}

Output: Updated network parameters θ

Initialize

- Replay buffer $\mathcal{D} \leftarrow \emptyset$
- Target network $Q_{\text{target}} \leftarrow Q_{\text{main}}$
- Exploration rate $\epsilon \leftarrow 1.0$

For each episode

- Reset environment: $s_0 \leftarrow \text{env.reset}()$
- foreach time step t do**
 - Step 1: Action Selection**
 - if** $\text{random}() < \epsilon$ **then**
 - Select random action $a_t \in \mathcal{A}$
 - else**
 - Compute Q -values from the main network:
 $Q(s_t, a) = \sum_{i=1}^{N_{\text{atoms}}} z_i \cdot p_i(s_t, a)$
 - Select action $a_t = \arg \max_a Q(s_t, a)$
 - Execute action a_t , observe reward r_t and next state s_{t+1}
 - Store transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ in replay buffer $\mathcal{D}$
 - Update exploration rate: $\epsilon \leftarrow \max(\epsilon \times \epsilon_{\text{decay}}, \epsilon_{\text{min}})$
- Step 2: Network Optimization**
 - Sample mini-batch $\mathcal{B}$ from prioritized replay buffer $\mathcal{D}$
 - foreach transition $(s_t, a_t, r_t, s_{t+1}, d_t) \in \mathcal{B}$ do**
 - Compute n -step return:
 $G_t^{(n)} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n Q_{\text{target}}(s_{t+n}, a^*)$
 - where $a^* = \arg \max_a Q_{\text{main}}(s_{t+n}, a)$
 - Compute TD error: $\delta = G_t^{(n)} - Q_{\text{main}}(s_t, a_t)$
 - Update priorities: $\mathcal{P}(i) \leftarrow |\delta|^\alpha + \epsilon$
 - Perform gradient descent step using loss:
 $\mathcal{L} = \mathbb{E}_{\mathcal{B}} [\delta^2]$
 - Periodically update target network: $Q_{\text{target}} \leftarrow Q_{\text{main}}$

return Updated network parameters θ

6 Results

We present VQR-DQN experimental setup, performance evaluation, and learning behavior analysis versus baseline (random assignment), DDQN, and Rainbow DQN using average rewards and normalized makespan reduction across varying HRAP complexity, assessing learning efficiency, stability, and final performance advantages.

6.1 Experimental Setup

Four HRAP configurations evaluate performance under varying complexity: 3 Officers - 2 Tasks - 2 Events (30-2T-2E), 4 Officers - 3 Tasks - 2 Events (40-3T-2E), 4 Officers - 3 Tasks - 3 Events (40-3T-3E), and 5 Officers - 4 Tasks - 4 Events (50-4T-4E). All methods were trained for 50,000 episodes under identical conditions, with the best-performing checkpoints evaluated across 200 testing episodes. Quantum simulations used TensorFlow Quantum with computations performed on IonQ Aria-1 quantum processing unit via IonQ quantum computing service.

6.1.1 Learning Curves. Figure 3 illustrates the learning curves over 50,000 training episodes for all configurations, demonstrating how each algorithm's performance evolves during the training process. The learning curves reveal that VQR-DQN generally achieved faster convergence and more stable learning compared to other approaches, particularly in the early stages of training. Learning speed and stability also vary with problem complexity. In simpler scenarios like 30-2T-2E, all algorithms achieve relatively quick and smooth convergence within the first 15,000 episodes. However, as the configuration complexity increases, particularly in 50-4T-4E, the convergence becomes notably slower and more gradual, with algorithms requiring nearly 30,000 to 40,000 episodes to stabilize. Despite this, VQR-DQN maintains more stable improvement compared to other algorithms, especially in later training stages.

6.1.2 Performance Evaluation. Table 1 shows the averaged rewards across 200 test episodes using the best checkpoint from training. In the simplest configuration (30-2T-2E), VQR-DQN achieved the most substantial improvement, showing a 26.8% increase in performance compared to the baseline, while DDQN and Rainbow DQN showed improvements of 13.1% and 19.8% respectively. This significant enhancement suggests that the quantum-enhanced feature extraction is particularly effective in capturing important patterns in simpler action spaces.

As the problem complexity increased in the 40-3T-2E configuration, VQR-DQN maintained its superior performance with a 23.7% improvement over the baseline, compared to DDQN's 15.1% and Rainbow DQN's 19.8%. The learning curves show that VQR-DQN not only achieved better final performance but also demonstrated more stable learning progression throughout the training process.

In more complex scenarios (40-3T-3E and 50-4T-4E), while the relative improvements were smaller due to increased problem difficulty, VQR-DQN still maintained its advantage. For the most complex configuration (50-4T-4E), VQR-DQN achieved a 10.1% improvement over the baseline, outperforming both DDQN (4.9%) and Rainbow DQN (7.2%). VQR-DQN consistently outperformed baseline random assignment, DDQN, and Rainbow DQN across all configurations, demonstrating that quantum-enhanced feature extraction provides robust, scalable advantages for resource allocation optimization, particularly in moderate complexity scenarios.

7 Impact of Topologies in VQCs for RL

We examine the correlation between expressibility and VQC topology performance in RL, specifically HRAP tasks. **Expressibility** quantifies how uniformly a circuit's random parameterizations explore state space through Kullback–Leibler divergence between circuit-produced states and uniform Haar-random distribution [10, 12]. **Entanglement** captures average circuit entanglement generation, measured via Meyer–Wallach (MW) measure [23], i.e., a global metric based on single-qubit reduced state purity ranging from 0 (no entanglement) to 1 (maximal multi-qubit entanglement). Alternative measures include Scott's multipartite quantifiers [31], averaging bipartite entanglement across all partitions.

Recent studies systematically evaluate entangler connectivities (Ring, Linear, Star, All-to-All) using these metrics. Both theoretical analyses and our experimental results indicate that Ring topologies provide rich expressibility and entanglement structure. In our

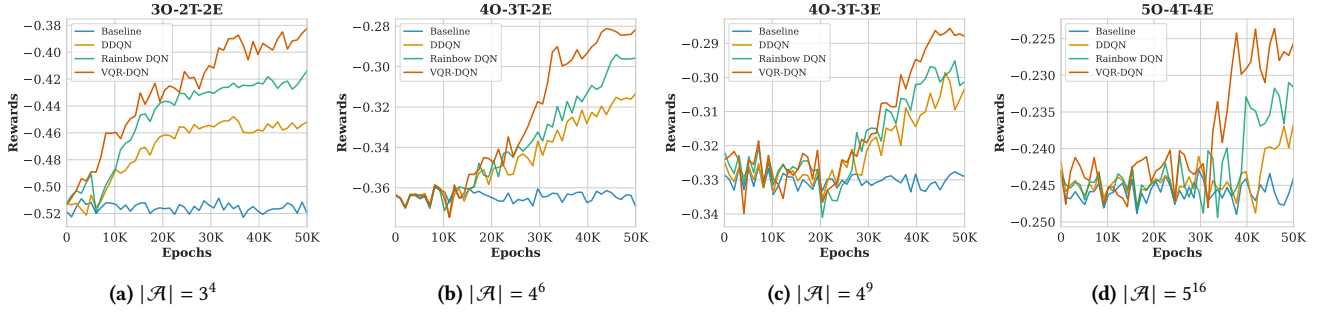


Figure 3: The learning curves for VQR-DQN and other algorithms in 50,000 episodes for different HRAP configurations (with their action space).

Table 1: Averaged rewards (Normalized makespan reduction%) over the baseline for VQR-DQN and other algorithms across 200 testing episodes, evaluated from the agent checkpoint with the highest training score. The best results are in bold.

Config.	$ \mathcal{A} $	Baseline	DDQN	Rainbow DQN	VQR-DQN (Ours)
30-2T-2E	3^4	-0.5225	-0.4539 ($\blacktriangle$ 13.1%)	-0.4189 ($\blacktriangle$ 19.8%)	-0.3823 ($\blacktriangle$ 26.8%)
40-3T-2E	4^6	-0.3689	-0.3132 ($\blacktriangle$ 15.1%)	-0.2957 ($\blacktriangle$ 19.8%)	-0.2815 ($\blacktriangle$ 23.7%)
40-3T-3E	4^9	-0.3316	-0.3032 ($\blacktriangle$ 8.6%)	-0.3012 ($\blacktriangle$ 9.2%)	-0.2872 ($\blacktriangle$ 13.4%)
50-4T-4E	5^{16}	-0.2488	-0.2366 ($\blacktriangle$ 4.9%)	-0.2309 ($\blacktriangle$ 7.2%)	-0.2236 ($\blacktriangle$ 10.1%)

Table 2: Averaged rewards (Normalized makespan reduction%) over the baseline for different topologies for VQR-DQN across 200 testing episodes in the configuration 30-2T-2E ($|\mathcal{A}| = 3^4$), evaluated from the agent checkpoint with the highest training score. The best results are in bold.

Algorithm	Rewards
Baseline	-0.5225
VQR-DQN + Linear	-0.4249 ($\blacktriangle$ 18.7%)
VQR-DQN + Star	-0.4514 ($\blacktriangle$ 13.6%)
VQR-DQN + Ring	-0.3823 ($\blacktriangle$ 26.8%)
VQR-DQN + All-to-All	-0.4103 ($\blacktriangle$ 21.5%)

HRAP environment, VQR-DQN with Ring topology achieved superior average rewards (Ring > All-to-All $\approx$ Linear $\gg$ Star) (see Table 2), aligning with [10] findings that Ring circuits exhibit the highest average expressibility and entanglement for given qubits and layers. This correlates with [16] observations of moderate-to-strong correlation between circuit expressibility and classification accuracy, and [33] noted “substantial improvement” in expressibility when two-qubit gates are arranged in Ring or All-to-All versus Linear topology.

Practically, Star topology entangles peripheral qubits only via the central qubit, creating a correlated entanglement structure, while Ring distributes entangling operations around the loop, promoting global entanglement [10]. Ring’s greater entangling reach yields higher MW scores and uniform state coverage, enabling quantum agents to represent complex HRAP policies. RL often requires modeling sequential or spatial correlations; Ring’s circular connectivity aligns with these structures, enabling efficient information flow and long-range dependency capture. However, the empirical connection between expressibility/entanglement metrics and VQC performance in RL tasks remains to be proven [33, 12].

8 Conclusion

We introduce VQR-DQN, a hybrid quantum-classical RL agent integrating Ring-topology VQCs into Rainbow DQN. VQR-DQN achieved superior HRAP performance, reducing normalized makespan by 26.8% versus random baseline and outperforming Double DQN

and Rainbow DQN by 4.9-13.4%. Ring-connected circuits consistently surpassed other topologies through broader entangling reach, aligning with theoretical expressibility and learning dynamics insights. These findings highlight quantum-enhanced feature extraction potential in complex decision-making, suggesting co-designed quantum architectures with RL objectives could yield further gains as quantum hardware advances.

Acknowledgement

This work has been supported by NSERC Discovery Grant No RGPIN 2025-00129.

References

- [1] Eva Andrés et al. 2022. On the use of quantum reinforcement learning in energy-efficiency scenarios. *Energies*, 15, 16, 6034.
- [2] Sana Anjum et al. 2024. Machine learning-based resource allocation algorithms for 6g networks. In *2024 2nd International Conference on Disruptive Technologies (ICDT)*. IEEE, 1086–1091.
- [3] James Adu Ansere et al. 2023. Quantum deep reinforcement learning for dynamic resource allocation in mobile edge computing-based iot systems. *IEEE Transactions on Wireless Communications*, 23, 6, 6221–6233.
- [4] Seyed Alireza Rahimi Azghadi et al. 2024. An energy-efficient lora iot system for water monitoring: lessons learned and use cases. In *2024 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 1–4.
- [5] Mina Azimi et al. 2013. Optimal allocation of human resources by using linear programming in the beverage company. *Universal Journal of Management and Social Sciences*, 3, 5, 48–54.
- [6] Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. 2020. Exploratory combinatorial optimization with reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 04, (Apr. 2020), 3243–3250.

- [7] Sana Bouajaja and Najoua Dridi. 2017. A survey on human resource allocation problem and its applications. *Operational Research*, 17, 339–369.
- [8] Maxime Bouton et al. 2019. Safe reinforcement learning with scene decomposition for navigating complex urban environments. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1469–1476.
- [9] Samuel Yen-Chi Chen et al. 2020. Variational quantum circuits for deep reinforcement learning. *IEEE access*, 8, 141007–141024.
- [10] Guilherme Ilário Correr et al. 2024. Characterizing randomness in parameterized quantum circuits through expressibility and average entanglement. *Quantum Science and Technology*, 10, 1, 015008.
- [11] Daoyi Dong et al. 2008. Quantum reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38, 5, 1207–1220.
- [12] Theodora-Augustina Drăgan et al. 2022. Quantum reinforcement learning for solving a stochastic frozen lake environment and the impact of quantum architecture choices. *arXiv preprint arXiv:2212.07932*.
- [13] Haoxuan Du and Lei Na. 2023. Automatic decision algorithm of human resource management system based on reinforcement learning. In *2023 International Conference on Internet of Things, Robotics and Distributed Computing (ICIRDC)*. IEEE, 277–282.
- [14] Zsolt Ercsey and Zoltán Kovács. 2024. Multicommodity network flow model of a human resource allocation problem considering time periods. *Central European Journal of Operations Research*, 32, 4, 1041–1059.
- [15] Matteo Hessel et al. 2018. Rainbow: combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32.
- [16] Thomas Hubregtsen et al. 2021. Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability. *Quantum Machine Intelligence*, 3, 1, 9.
- [17] Bassem Jarboui et al. 2008. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195, 1, 299–308.
- [18] Sofiene Jerbi et al. 2021. Parametrized quantum policies for reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 28362–28375.
- [19] Sekione Reward Jeremiah et al. 2024. Digital twin-assisted resource allocation framework based on edge collaboration for vehicular edge computing. *Future Generation Computer Systems*, 150, 243–254.
- [20] Petar Kormushev et al. 2013. Reinforcement learning in robotics: applications and real-world challenges. *Robotics*, 2, 3, 122–148.
- [21] Timothy P Lillicrap et al. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- [22] Neda Manavizadeh et al. 2013. A simulated annealing algorithm for a mixed model assembly u-line balancing type-i problem considering human efficiency and just-in-time approach. *Computers & industrial engineering*, 64, 2, 669–685.
- [23] David A Meyer and Nolan R Wallach. 2002. Global entanglement in multipartite systems. *Journal of Mathematical Physics*, 43, 9, 4273–4278.
- [24] Nico Meyer et al. 2022. A survey on quantum reinforcement learning. *arXiv preprint arXiv:2211.03464*.
- [25] Volodymyr Mnih et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518, (Feb. 2015), 529–533.
- [26] Özcan Mutlu et al. 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-ii. *Computers & Operations Research*, 40, 1, 418–426.
- [27] Phong Nguyen et al. 2021. Can reinforcement learning solve a human allocation problem? *Bridging the Gap Between AI Planning and Reinforcement Learning (PRL) Workshop, ICAPS 2021*.
- [28] Truong Thanh Hung Nguyen et al. 2024. Temporal point processes for business process monitoring. *Quy Nhon University Journal of Science*. doi:10.52111/qnjs.2024.18501.
- [29] Ciprian Paduraru et al. 2021. Task distribution and human resource management using reinforcement learning. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. IEEE, 96–101.
- [30] Xue Bin Peng et al. 2017. Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36, 4, 1–13.
- [31] Andrew J Scott. 2004. Multipartite entanglement, quantum-error-correcting codes, and entangling power of quantum evolutions. *Physical Review A—Atomic, Molecular, and Optical Physics*, 69, 5, 052330.
- [32] David Silver et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.
- [33] Sukin Sim et al. 2019. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2, 12, 1900070.
- [34] Andrea Skolik et al. 2022. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *Quantum*, 6, 720.
- [35] Rafael Sotelo. 2023. Quantum in consumer technology. *IEEE Consumer Electronics Magazine*, 12, 5, 4–7.
- [36] Anh Son Ta and Thi Thuy Nguyen. 2024. Solving resource allocation problem in wifi network by dantzig-wolfe decomposition algorithm. *JST: Smart Systems and Devices*, 34, 1, 9–15.
- [37] Nguyen Duy Tan et al. 2024. Optimization and inventory management under stochastic demand using metaheuristic algorithm. *Plos one*, 19, 1, e0286433.
- [38] Hado Van Hasselt et al. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30.
- [39] Mariona Vila and Jordi Pereira. 2014. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers & Operations Research*, 44, 105–114.
- [40] Ziyu Wang et al. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1995–2003.
- [41] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning*, 8, 279–292.
- [42] Hairun Xu et al. 2025. Quantum reinforcement learning for real-time optimization in electric vehicle charging systems. *Applied Energy*, 383, 125279.
- [43] Tong Zhang et al. 2021. Udarmf: an underwater distributed and adaptive resource management framework. *IEEE Internet of Things Journal*, 9, 10, 7196–7210.

A Appendix

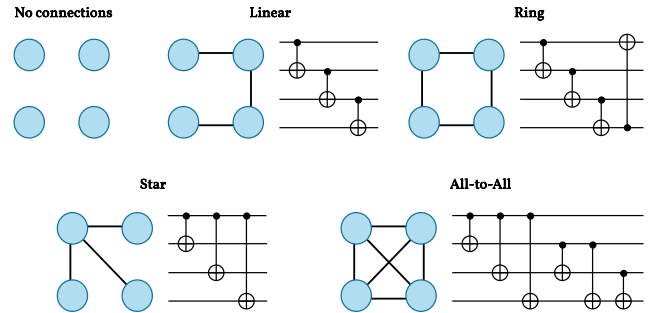


Figure 4: Graphs of the topologies observed in different quantum computer architectures.

In this section, we introduce fundamental terminologies essential for understanding VQCs.

Topologies. (Figure 4) to the different graph topologies related to each of the connectivities between qubits that can be performed in quantum hardware. Common topologies include:

- **Linear:** Qubits lie in a chain, each entangled only with immediate neighbors.
- **Ring:** Arranged in a circle, each qubit is entangled with two neighbors.
- **Star:** A central qubit directly connects to all peripheral qubits, and the latter do not interconnect.
- **All-to-All:** Every qubit can entangle with every other qubit.

Ansätze. refers to the specific circuit structure implemented within a given topology:

- **Parameterized gates:** Rotational operations (RX, RY, RZ) with trainable angles that can be optimized through classical algorithms.
- **Entangling gates:** Fixed operations (typically CNOT) that create quantum correlations between qubits according to the chosen topology.
- **Layered structure:** Repeating blocks of parameterized and entangling gates that increase circuit depth and expressivity.
- **Hardware-efficient ansatz:** Circuit designs that minimize the number of gates while maintaining computational power.