QFOR: A Fidelity-aware Orchestrator for Quantum Computing Environments using Deep Reinforcement Learning

HOA T. NGUYEN, Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia
MUHAMMAD USMAN, School of Physics, The University of Melbourne, Australia and Data61, CSIRO, Australia
RAJKUMAR BUYYA, Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia

Quantum cloud computing enables remote access to quantum processors, yet the heterogeneity and noise of available quantum hardware create significant challenges for efficient resource orchestration. These issues complicate the optimisation of quantum task allocation and scheduling, as existing heuristic methods fall short in adapting to dynamic conditions or effectively balancing execution fidelity and time. Here, we propose QFOR, a **Q**uantum **F**idelity-aware **O**rchestration of tasks across heterogeneous quantum nodes in cloud-based environments using Deep **R**einforcement learning. We model the quantum task orchestration as a Markov Decision Process and employ the Proximal Policy Optimisation algorithm to learn adaptive scheduling policies, using IBM quantum processor calibration data for noise-aware performance estimation. Our configurable framework balances overall quantum task execution fidelity and time, enabling adaptation to different operational priorities. Extensive evaluation demonstrates that QFOR is adaptive and achieves significant performance with 29.5-84% improvements in relative fidelity performance over other deep reinforcement learning and heuristic baselines. Furthermore, it maintains comparable quantum execution times, contributing to cost-efficient use of quantum computation resources.

CCS Concepts: • **Computer systems organization → Quantum computing**; **Cloud computing**.

Additional Key Words and Phrases: quantum cloud computing, quantum cloud orchestration, reinforcement learning for quantum, quantum resource management

ACM Reference Format:
Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya. 2026. QFOR: A Fidelity-aware Orchestrator for Quantum Computing Environments using Deep Reinforcement Learning. 1, 1 (February 2026), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The rapid advancement of quantum computing promises to solve computationally intractable problems across critical technology domains, including cryptography [28], drug discovery [41], optimisation [19], and machine learning [2]. However, given the significant challenges associated with operating physical quantum hardware, such as stringent

Authors' Contact Information: Hoa T. Nguyen, thanhhoan@student.unimelb.edu.au, Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, Victoria, Australia; Muhammad Usman, School of Physics, The University of Melbourne, Parkville, Victoria, Australia and Data61, CSIRO, Clayton, Victoria, Australia, musman@unimelb.edu.au; Rajkumar Buyya, Quantum Cloud Computing and Distributed Systems (qCLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville, Victoria, Australia, rbuyya@unimelb.edu.au.

Manuscript submitted to ACM

environmental requirements and high costs, Quantum Cloud Computing (QCC) has emerged as an access paradigm [22]. QCC platforms, offered by major providers like IBM Quantum, Amazon Braket, Google Cloud, and Microsoft Azure, democratize access to Quantum Processing Units (QPUs) in a Quantum-as-a-Service (QaaS) model. This allows users to execute quantum applications remotely without the need for on-premise hardware.

Notably, current QPUs are not fully fault-tolerant processors and operate within the constraints of the Noisy Intermediate-Scale Quantum (NISQ) era [29], and executing quantum applications on them is a hybrid process, involving both classical and quantum execution. For instance, Variational Quantum Algorithms (VQAs) necessitate a classical optimisation step for convergence [21]. Even fully quantum algorithms require classical transpilation to be compatible with the qubit topology and native gate set of the targeted QPU. Furthermore, realising quantum advantage requires seamless integration with classical high-performance computing (HPC) infrastructure to form hybrid quantum-classical systems [1, 34]. These hybrid systems leverage classical HPC resources for pre-processing, optimisation, and post-processing while utilising quantum processing units (QPUs) for quantum-specific computations. Figure 1 presents a high-level architecture of a quantum-classical hybrid cloud system, wherein user requests are routed via an API gateway to a corresponding service or application deployment at the middleware layer. Classical tasks for pre-processing and post-processing are queued and dispatched to CPUs and GPUs, while quantum tasks are orchestrated by a quantum task orchestrator component (for example, our proposed QFOR orchestrator in this work), which manages quantum processing units (QPUs) through dedicated queues and APIs.

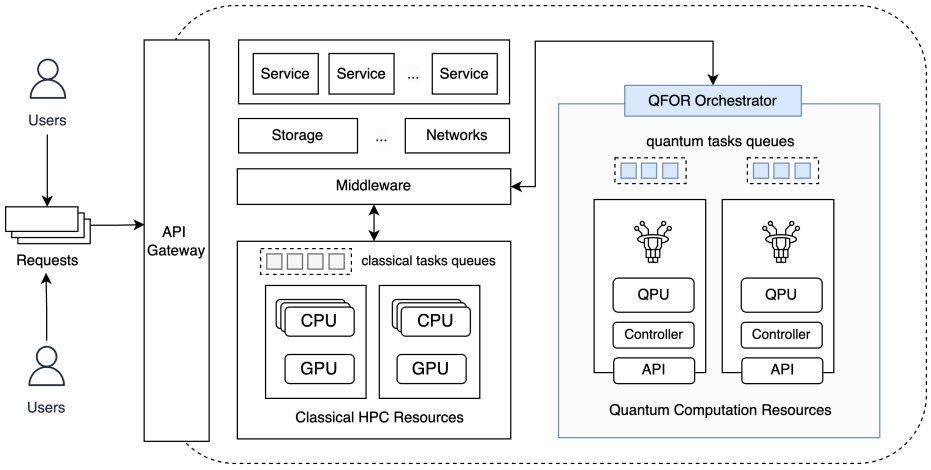


Fig. 1. High-level view of a Quantum-HPC system in the cloud-based environment. The QFOR Orchestrator (proposed in this work) manages quantum task scheduling and coordination across quantum processing units (QPUs).

In these hybrid quantum cloud computing environments, effective quantum task orchestration is crucial for three key reasons. First, quantum computation resources exhibit extreme scarcity as quantum hardware development is still in its early stages [22]. Second, quantum resources incur costs significantly higher than equivalent classical compute time, making efficient utilisation economically critical. For example, each minute of quantum execution on IBM Quantum hardware costs \$96 USD[1], while each hour of IonQ device reservation on Amazon Braket costs \$7,000 USD[2] (as of

[1]https://www.ibm.com/quantum/pricing
[2]https://aws.amazon.com/braket/pricing/

June 2025). Third, quantum fidelity degradation due to poor scheduling can invalidate entire hybrid computations, regardless of classical processing quality [10, 14]. Current quantum hardware exhibits significant variability across multiple dimensions that directly impact quantum task execution [22]. Multiple QPU technologies exist, including superconducting qubits, trapped ions, and neutral atoms, each with distinct performance and noise characteristics. Even within the same technology, different QPU models possess varied qubit layouts, native gate sets, and qubit connectivity. Architecturally, devices differ in qubit connectivity constraints how quantum circuits can be efficiently mapped and executed. Additionally, gate durations and error rates vary not only between devices but also over time due to calibration cycles and environmental factors [4, 33]. Device availability is another critical consideration, as queue times can fluctuate based on user demand and system maintenance [32]. These factors collectively introduce substantial heterogeneity and uncertainty into the quantum cloud environment. As a result, effective orchestration must be both fidelity-aware and dynamically select quantum computation nodes that optimise for execution reliability while maintaining efficiency with reasonable execution time. This orchestration approach is essential for harnessing the potential of quantum cloud resources and ensuring consistent, high-quality results and cost efficiency for users.

Despite the increasing interest in QCC, existing research in quantum cloud resource management exhibits several critical gaps, particularly concerning heterogeneous quantum computation resources and noise-aware execution fidelity [22]. Traditional resource management approaches fall into two categories, both with critical limitations. Traditional HPC scheduling algorithms excel at managing classical computation resources but can struggle to account for quantum-specific constraints such as fidelity optimisation, decoherence, and calibration-dependent performance [1, 8]. Conversely, quantum-specific heuristics [15, 24, 25, 32] are inherently limited in their ability to adapt to the dynamic and uncertain nature of quantum cloud environments. Recent AI-driven approaches, such as deep reinforcement learning (DRL), have shown promise for addressing this problem with several successful cases in the classical cloud-edge [9, 11] and high-performance computing domain [38, 40], as it is well-suited for sequential decision-making in environments with incomplete information and stochastic dynamics. However, DRL approaches for quantum cloud orchestration remain limited in scope. Existing works mainly focused on completion time [23] and device allocation [16] or overall fidelity of the targeted quantum device [20]. To our knowledge, no existing work considers the trade-off between circuit execution fidelity, time, and complexity of quantum tasks using DRL-based approaches with real quantum circuit workload.

Furthermore, as quantum cloud resources are scarce and limited, designing and evaluating resource management in a practical environment is extremely challenging. Therefore, simulation frameworks [20, 24, 26] for modelling and simulating quantum cloud computing environments are essential for this research area. However, existing approaches mainly focus on high-level metrics for the performance estimation of quantum tasks, such as quantum volume [7] and circuit layer operation per second (CLOPS) [37]. Although this approach is promising, it lacks the adaptability to different structures of quantum circuits and their transpilation to different qubit topologies of quantum hardware. Indeed, specific information on gate errors and durations of individual qubits in quantum hardware can be used to enhance the estimation process. Thus, there is a clear need for resource estimation that systematically explores this fidelity-runtime tradeoff to aid optimal quantum resource orchestration.

To address the aforementioned challenges, we propose QFOR, a **Q**uantum **F**idelity-aware **O**rchestrator that uses a Deep **R**einforcement learning-based approach to optimise the overall performance of task orchestration in quantum cloud environments. QFOR models quantum task placement as a Markov Decision Process and employs Proximal Policy Optimisation (PPO) [36] to learn adaptive policies that balance overall execution fidelity and time of quantum tasks.

The major contributions and novelty of our work are:

- We propose a novel deep reinforcement learning-based task orchestration framework for quantum computing in cloud-based environments. Our method considers the critical trade-off between execution fidelity and quantum execution time, with a primary focus on maximising fidelity-aware overall performance.
- Our approach employs a systematic quantum task execution and fidelity estimators, and mimics the execution of noisy devices based on calibration data and actual quantum circuit properties, enabling more rigorous quantum cloud modelling and simulation of quantum hardware behaviour and improved orchestration decisions.
- We provide configurable orchestration objectives that balance execution fidelity and latency, accounting for circuit complexity and task priority. Extensive evaluation demonstrates that QFOR is flexible and adaptive, achieving 29.5–84% improvements in relative fidelity compared to other deep reinforcement learning and heuristic baselines, while maintaining comparable quantum execution times, thus supporting cost-efficient quantum resource usage.

The rest of the paper is organised as follows: Section 2 reviews existing works related to our study. Section 3 describes the system model and formulates the problems of quantum task orchestration in heterogeneous quantum cloud computing environments. We provide details of the methodology and design of the QFOR framework in Section 4. Then, Section 5 describes the evaluation study of our proposed framework, followed by a further discussion on the results. Finally, we conclude the paper with key insights, limitations, and future directions in Section 6.

2 Related Work

In this section, we review existing literature on quantum resource management and orchestration, categorising approaches by their methodology and highlighting critical gaps that our work addresses. Table 1 provides a comprehensive comparison across key technical dimensions.

Table 1. Overall comparison of related works on quantum cloud task orchestration and resource management problem. *(H: Heuristic, DRL: Deep Reinforcement Learning, N/A: Not available, ◐: Partially Addressed/Described, Env: Environments, E: Emulation with real quantum circuit compilation and execution, S: Simulation with circuit feature or synthetic data)*

Works	**Quantum Tasks**		**Quantum Nodes**		**Method**	**Optimization config**			**Env**
	Real Circuit	*Dataset/Task Generator*	*Noise Aware*	*Qubit*		*Fidelity*	*Time*	*Weighted*	
QFaaS [25]	✓	Qiskit	✓	7-127	H	◐	◐	✗	E
Ravi et al. [32]	✓	Qiskit	✓	1-65	H	✓	◐	✗	S
iQuantum [24]	✓	MQTBench	✗	27-127	H	✗	✓	✗	S
Qonductor [10]	✓	MQTBench	✓	27	H	✓	◐	✗	E
QuSplit [15]	✓	Qiskit (VQE)	✓	N/A	H	✓	✗	✗	S
DRLQ [23]	✓	MQTBench	✗	27-127	DRL	✗	✓	✗	S
Moirai [16]	✓	Qiskit	✗	5-7	DRL	✗	✓	✗	E
Luo et al. [20]	✗	Random data	✓	127	DRL	✓	◐	✗	S
QFOR	✓	**MQTBench**	✓	**27-127**	**DRL**	✓	✓	✓	**E**

Early papers in quantum cloud orchestration have primarily relied on heuristic approaches that extend classical scheduling paradigms to quantum environments. Ravi et al. [32] introduced an adaptive job and resource management for quantum clouds, focusing on fidelity optimisation through device selection. However, their approach relies on predetermined heuristics and statistical analysis that can struggle to adapt to temporal variations in quantum device performance. QFaaS [25] is one of the first serverless quantum computing frameworks, establishing a Function-as-a-Service model for quantum task execution. While QFaaS demonstrates practical hybrid quantum-classical integration

across multiple cloud providers, it employs primarily heuristic models for resource allocation decisions based on execution priority (speed and accuracy) using Quantum Volume (QV) [7], CLOPS [37], and queue metrics without considering dynamic fidelity-runtime tradeoffs. Pioneering the discrete-event quantum cloud modelling and simulation, iQuantum [24] and QSimPy [26] provide comprehensive toolkits for quantum cloud resource management design and evaluation, but have not fully considered noise-aware modelling. This gap is also one of the key motivations for QFOR to address, providing an emulation approach to mimic the noisy-quantum hardware in practical quantum cloud environments. Qonductor [10] represents one of the most sophisticated heuristic frameworks, offering a hybrid quantum-classical orchestration. Despite introducing important concepts like hybrid resource estimation, Qonductor relies on heuristic scheduling policies and a prediction model based on historical data that can be challenging to adapt to dynamic quantum environments. Focusing only on quantum optimisation applications, QuSplit [15] focuses on optimising fidelity and throughput through job splitting using a genetic algorithm. However, the limitation of heuristic approaches lies in their ability to adapt to the dynamic and stochastic nature of quantum cloud environments. As quantum hardware and the characteristics of quantum tasks evolve, static scheduling policies become harder to adapt to, requiring adaptive learning-based approaches.

Recent research has begun exploring deep reinforcement learning as a solution to quantum orchestration challenges, demonstrating significant improvements over heuristic baselines while revealing important limitations. DRLQ [23] pioneered the use of deep reinforcement learning [12] for quantum task scheduling, demonstrating significant improvements over heuristic baselines in terms of completion time. Similarly, Moirai [16] employed policy gradient methods within the OpenWhisk framework to schedule quantum circuits on small-scale quantum devices. However, neither DRLQ nor Moirai incorporated comprehensive noise-awareness in the decision-making process. Recently, Luo et al. [20] employed a DRL approach in a simulated environment to maximise targeted device fidelity on different 127-qubit quantum nodes. However, their work relied on CLOPS-based estimation for quantum execution with synthetic random data for the training and evaluation, rather than real quantum circuits, and did not consider the execution time factor in the DRL policy design to fully address the tradeoff between time and fidelity of task execution.

As summarised in Table 1, our work addresses several limitations in the existing approaches. Existing approaches lack comprehensive performance modelling that integrates circuit features, device calibration data, and noise-aware fidelity and execution time estimation for improving scheduling decisions. Besides, current DRL-based methods operate on limited-scale systems or synthetic random data in simulated environments rather than emulating the execution with quantum circuit compilation and execution, which have limitations in practical applicability to quantum cloud environments. Our work provides a comprehensive, deep reinforcement learning-based framework that considers noise-aware performance modelling using device calibration data and configurable optimisation with evaluation on realistic quantum circuit workloads from a well-known quantum circuit dataset (MQTBench [31]). Our work aims to contribute valuable approaches and insights for future works in quantum cloud orchestration, enabling learning-driven resource management that adapts to the dynamic, heterogeneous nature of NISQ-era quantum computing environments while optimising for practical deployment requirements.

3 System Model and Problem Formulation

This section presents our system model and problem formulation for quantum cloud orchestration, focusing on fidelity-aware performance metrics.

3.1 Quantum Task Model

In quantum cloud environments, a quantum task (QTask) represents the unit of quantum computation requiring adaptive resource orchestration to optimise the performance. A QTask can comprise one or multiple quantum circuits that need to be executed with specific qubit requirements, gate operations, and circuit depth. In the context of this work, we consider each QTask as a single independent quantum circuit. These QTasks can encompass different quantum algorithms, each with distinct characteristics.

Let $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ denote a sequence of quantum tasks arriving for execution. Each task $\tau_i \in \Gamma$ is characterized by the following properties:

$$\tau_i = \left(a_i, q_i, d_i, s_i, g_i^{(1)}, g_i^{(2)}, C_i\right) \tag{1}$$

where:

- $a_i \in \mathbb{R}_{\geq 0}$: arrival time of the task into the system.
- $q_i \in \mathbb{N}$: number of qubits required for execution.
- $d_i \in \mathbb{N}$: circuit depth, i.e., the longest gate dependency path.
- $s_i \in \mathbb{N}$: number of shots (i.e., repetition of the execution)
- $g_i^{(1)} \in \mathbb{N}$: total number of single-qubit gates.
- $g_i^{(2)} \in \mathbb{N}$: total number of two-qubit gates.
- C_i: the quantum circuit representation as a Directed Acyclic Graph (DAG).

Each circuit C_i is modelled as a DAG $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$, where $\mathcal{V}_i$ is the set of all quantum operations (gates), and $\mathcal{E}_i \subseteq \mathcal{V}_i \times \mathcal{V}_i$ represents directed edges indicating gate dependencies.

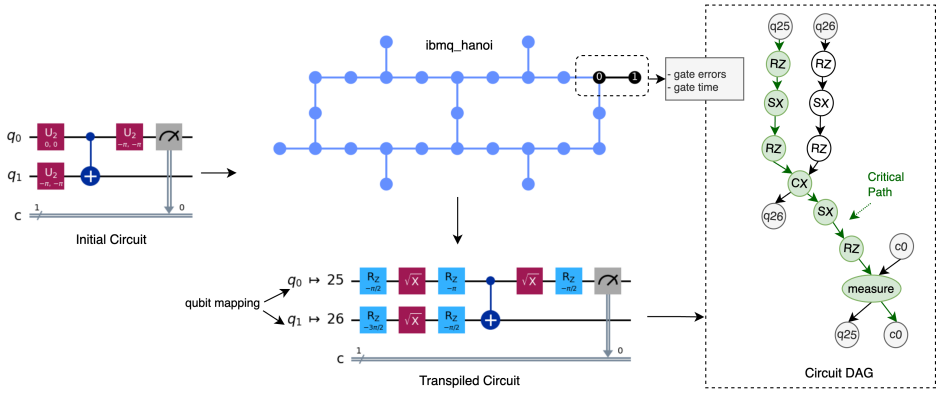


Fig. 2. Example of an initial quantum circuit and its transpilation, qubit mapping to a 27-qubit QNode (ibmq_hanoi), and DAG of the transpiled circuit with the critical path illustration.

Example of a quantum circuit within a QTask and its transpilation, mapping and DAG representation are illustrated in Figure 2. The DAG representation enables critical path analysis for the estimation of QTask execution time and the execution fidelity based on the calibration data of QNodes. Based on the current state of typical quantum cloud environments [22], we assume each task requires exclusive access to a set of qubits on a quantum processor, and no preemption or interruption occurs during task execution. The orchestration focuses exclusively on quantum circuit execution, which dominates in terms of resource sensitivity compared to classical resources.

3.2 Quantum Computation Resource Model

The quantum cloud infrastructure consists of heterogeneous quantum nodes (QNodes) with one or multiple quantum processing units (QPUs) with distinct performance characteristics that directly impact scheduling decisions. In the context of this work, we consider a single QPU per QNode to reflect the current state of available quantum hardware. Each QNode exhibits unique hardware specifications and properties, which must be taken into account during scheduling to achieve reliable and efficient quantum task execution.

Let the set of quantum resources be denoted by: $\mathcal{N} = \{n_1, n_2, \ldots, n_M\}$, where each node $n_j \in \mathcal{N}$ is defined as:

$$n_j = \left(q_j, \mathcal{G}_j, \mathcal{D}_j, \mathcal{E}_j, \rho_j(t)\right) \tag{2}$$

where:

- $q_j = |\mathcal{Q}_j| \in \mathbb{N}$: number of physical qubits and $\mathcal{Q}_j$ is the set of all available qubits at QNode n_j.
- $\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j)$ is the qubit connectivity graph, where V_j is the set of all qubits and E_j is the set of all edges connecting these qubits.
- $\mathcal{D}_j : \mathcal{O}_j \times \mathcal{Q}_j \to \mathbb{R}_{>0}$ maps available gates $\mathcal{O}_j$ and qubits $\mathcal{Q}_j$ to corresponding gate execution durations
- $\mathcal{E}_j : \mathcal{O}_j \times \mathcal{Q}_j \to [0, 1]$ maps all available gates $\mathcal{O}_j$ and qubits $\mathcal{Q}_j$ to corresponding error probabilities.
- $\rho_j(t)$: other dynamic state of the QNode at time t, such as next available time and queuing information.

This abstraction allows us to capture both static and time-dependent dynamics of each quantum node. These features are critical for orchestrator decisions in dynamic workload settings and heterogeneous quantum cloud environments.

3.3 Fidelity-aware Orchestration Performance Model

For quantum cloud orchestration in the NISQ era, there are two critical metrics that indicate the performance of the orchestration decision: execution fidelity and time. First, execution fidelity can be considered as one of the most critical factors. Quantum execution exhibits high sensitivity to noise, where small fidelity degradations can render results meaningless regardless of execution speed [37]. Poor device selection as well as poor selection of qubits on the targeted device can reduce algorithmic success probability, necessitating multiple re-executions that far exceed any time savings from faster scheduling. Second, quantum task execution time in the quantum node is also critical, as quantum resources are scarce and extremely expensive [22]. Besides, quantum states decay exponentially with time, making execution time a fundamental physical constraint rather than merely an optimisation preference [13]. Longer execution sequences suffer increased error accumulation, creating a direct coupling between execution time and fidelity. Therefore, we define the orchestration performance metric that mainly focuses on the fidelity of the execution, while maintaining the execution time and considering the complexity of the task that needs to be executed, as well as how good it compares to other available decisions.

3.3.1 Fidelity Performance Score. The execution fidelity performance score $\mathcal{F}_{i,j}$ is the key objective in this work and captures the comprehensive quality of executing task τ_i on node n_j through three complementary components:

For task τ_i assigned to node n_j, the base execution fidelity $F_{i,j}$ is approximately estimated as:

$$F_{i,j} = \prod_{g \in C_i'} \left(1 - \mathcal{E}_j(g, \vec{q}_g)\right) \tag{3}$$

where C_i' represents the transpiled circuit and $\vec{q}_g$ are the assigned physical qubits.

The base fidelity score R_{rf} normalises this against expected fidelity performance $F'_{i,j}$ based on the average gate errors of all available quantum nodes, preventing hardware-specific biases while rewarding above-expected performance.

$$R_{rf} = \frac{F_{i,j}}{F'_{i,j}} \tag{4}$$

In quantum cloud environments, simpler circuits (i.e., those with shallow depth and fewer gates) tend to achieve higher fidelity due to reduced exposure to noise. As a result, an orchestration policy that solely maximises fidelity may develop a bias toward such circuits, systematically deprioritising more complex tasks. To address this issue, we introduce a small complexity bonus R_{cb}, which encourages the orchestrator to also consider more complex circuits:

$$R_{cb} = w_d \cdot \frac{d_i}{D_{max}} + w_g \cdot \sum \frac{g_i}{G_{max}} \tag{5}$$

where d_i is circuit depth, $\sum g_i$ is total gate count in the circuit of the scheduled QTask, w_d and w_g are adjustable weights, and D_{max}, G_{max} are normalisation bounds for the maximum depth and gates of the circuit. This ensures fairness and task diversity within the orchestration policy while remaining computationally efficient and easy to integrate into learning-based decision frameworks. We also introduce a ranking bonus R_{rb} that captures the relative quality of selecting a specific quantum node compared to all available options for a given task to ensure that the orchestration policy not only aims for high absolute fidelity but also makes competitively optimal decisions based on the current system state. It is defined as:

$$R_{rb} = \frac{F_{i,j} - F_{worst}}{F_{best} - F_{worst}} \tag{6}$$

where F_{best} and F_{worst} represent the highest and lowest achievable fidelity across all nodes for task τ_i.

The comprehensive execution fidelity performance score (or relative fidelity) combines these components:

$$\mathcal{F}_{i,j} = \alpha_1 \cdot R_{rf} + \alpha_2 \cdot R_{cb} + \alpha_3 \cdot R_{rb} \tag{7}$$

where α_1, α_2, and α_3 are configurable weights and were set by default, and $\alpha_1 = 0.8, \alpha_2 = \alpha_3 = 0.1$, emphasising the key focus of our orchestration on execution fidelity while maintaining circuit complexity and decision ranking awareness. By aggregating these complementary components, $\mathcal{F}_{i,j}$ provides a robust fidelity-based performance measure. It supports fair and adaptive orchestration across diverse workloads and heterogeneous quantum hardware, and serves as a principled reward signal in reinforcement learning-based policy optimisation of this work.

3.3.2 Time Penalty Score. The time penalty score $\mathcal{T}_{i,j}$ captures the temporal cost of quantum task execution, encompassing both waiting and actual quantum execution time. For task τ_i assigned to node n_j, the quantum execution time is estimated by analysing the critical path of the transpiled circuit:

$$T_{i,j}^{exec} = s_i \cdot \sum_{g \in \mathrm{CP}(C'_i)} \mathcal{D}_j(g, \vec{q}_g) \tag{8}$$

where $\mathrm{CP}(C'_i)$ represents the critical path (longest execution sequence) of the transpiled circuit, and $\mathcal{D}_j(g, \vec{q}_g)$ is the gate execution duration along the critical path, s_i is the number of shots (execution iterations). An example of the longest path of a quantum circuit is illustrated in Figure 2. The total completion time includes queuing delays of QTask until it can be executed at the targeted QNode $\mathcal{T}_{i,j} = T_{i,j}^{wait} + T_{i,j}^{exec}$ and is normalised based on the maximum completion time bound to enable stable policy training.

3.4 Problem Formulation

Given the fidelity performance score $\mathcal{F}_{i,j}$ and time penalty score $\mathcal{T}_{i,j}$ defined above, we formulate the quantum cloud orchestration problem as a sequential decision-making process that maximises the combined orchestration performance across all quantum tasks.

The quantum cloud orchestration problem can be formally stated as follows: Given a sequence of quantum tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ arriving dynamically in a quantum cloud environment with heterogeneous quantum nodes $\mathcal{N} = \{n_1, n_2, \ldots, n_M\}$, find an optimal orchestration policy $\pi : \Gamma \rightarrow \mathcal{N}$ that assigns each task τ_i to an appropriate quantum node $n_j = \pi(\tau_i)$ to maximize the overall orchestration performance. For each assignment decision $\pi(\tau_i)$, the overall orchestration performance is quantified by combining the fidelity performance score and time penalty score, with a negative value of time score indicating the secondary goal to minimise the time penalty:

$$\mathcal{P}_{i,\pi(\tau_i)} = \mathcal{F}_{i,\pi(\tau_i)} - \beta \cdot \mathcal{T}_{i,\pi(\tau_i)} \tag{9}$$

where $\mathcal{F}_{i,\pi(\tau_i)}$ represents the relative fidelity performance score, β is the configurable time penalty weight and $\mathcal{T}_{i,\pi(\tau_i)}$ represents the time penalty, ensuring that higher fidelity and lower execution time both contribute positively to the overall performance score. The primary optimisation objective is to maximise the cumulative orchestration performance across all quantum tasks, which can be defined as follows:

$$\max_{\pi} \sum_{i=1}^{N} \mathcal{P}_{i,\pi(\tau_i)} = \max_{\pi} \sum_{i=1}^{N} \left[\mathcal{F}_{i,\pi(\tau_i)} - \beta \cdot \mathcal{T}_{i,\pi(\tau_i)}\right] \tag{10}$$

subject to:

$$C1 : Size(\pi(\tau_i)) = 1, \forall \pi(\tau_i) \in \{1, \ldots, M\} \tag{11}$$

$$C2 : q_{\tau_i} \leq q_{\pi(\tau_i)}, \forall i \in \{1, \ldots, N\} \tag{12}$$

$$C3 : C_{\tau_i} \hookrightarrow \mathcal{G}_{\pi(\tau_i)}, \forall i \in \{1, \ldots, N\} \tag{13}$$

where N is the total number of quantum nodes, M is the total number of tasks that need to be scheduled. $C1$ shows that each QTask will be allocated to exactly one QNode at a time, $C2$ indicates that the number of qubits in the targeted QNode needs to be larger than or equal to the number of qubits required by the allocated QTask, and $C3$ implies that the quantum circuit of the QTask can be mapped to the QNode through the transpilation process. This optimisation problem exhibits several challenges. First, quantum node characteristics $(\mathcal{D}_j, \mathcal{E}_j)$ vary with calibration cycles, and queue states $\rho_j(t)$ change with task arrivals and completions, requiring adaptive decision-making capabilities. Second, the discrete assignment decisions combined with non-linear relationships between circuit characteristics and performance metrics create a combinatorial optimisation problem. Furthermore, the sequential arrival of quantum tasks with unknown future characteristics necessitates an adaptive optimisation without complete future information. The tension between fidelity maximisation and time minimisation requires sophisticated balancing strategies that adapt to different priorities.

Given these complexities, traditional heuristic approaches cannot effectively navigate the dynamic trade-offs inherent in quantum cloud orchestration. Therefore, we propose a deep reinforcement learning approach that models this problem as a Markov Decision Process, enabling the learning of adaptive orchestration policies that can balance fidelity and time objectives with a set of configurable weights that can be adjusted based on the priority of the orchestration. The sequential nature of the decision-making process, combined with the need for adaptive policy learning, makes

reinforcement learning particularly well-suited for this orchestration challenge, which is widely used effectively for resource management in the classical computing environments [5, 9, 11].

4 QFOR Framework and Technique

4.1 Main components and Design

The QFOR framework implements an adaptive orchestration system that applies deep reinforcement learning to optimise quantum task placement in heterogeneous cloud environments. Figure 3 illustrates the framework architecture, comprising six integrated components that collectively enable adaptive, fidelity-aware orchestration decisions.

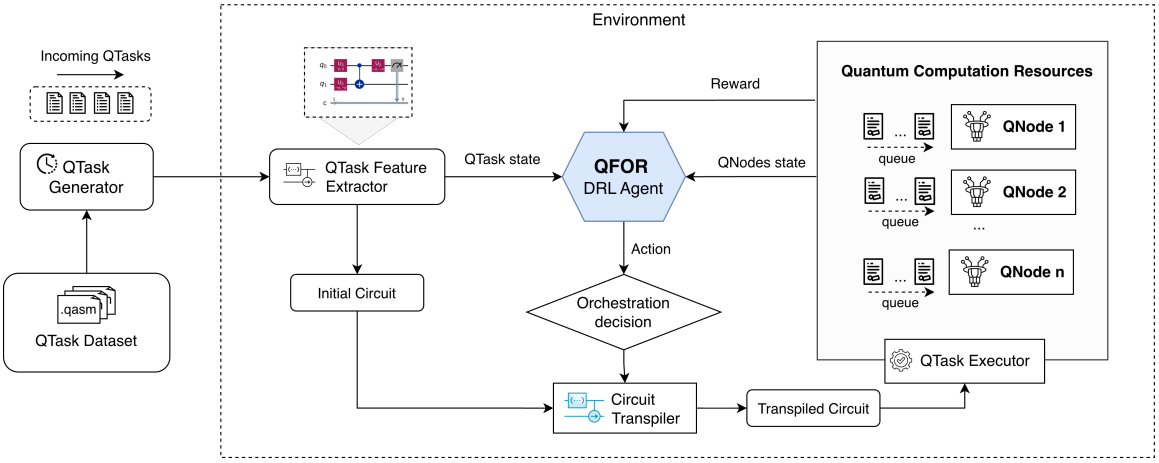


Fig. 3. Overview of the main components of the QFOR framework

(1) *QTask Generator and Dataset*: The QTask Generator serves as the workload interface, consuming quantum circuit representations from standardised datasets (for example, MQTBench [31]) in OpenQASM format [6]. This component simulates realistic quantum cloud workloads by generating tasks with diverse complexity characteristics—varying qubit requirements, circuit depths, and algorithmic patterns. The generator also supports configurable quantum task arrival patterns to model different cloud workload scenarios.

(2) *QTask Feature Extractor:* This component analyses quantum circuits to extract critical features, including qubit count, circuit depth, gate statistics (single-qubit, two-qubit gate, measurement counts), and circuit structure. These features are normalised and encoded into a concise representation that characterises the computational requirements and complexity of each QTask, enabling the DRL agent to make informed scheduling decisions based on circuit characteristics.

(3) *Deep Reinforcement Learning Agent*: The key component that implements the DRL-based decision-making policy. It observes the current state of quantum tasks and available quantum nodes, processes this information through a DRL agent, and produces orchestration decisions that assign tasks to appropriate quantum nodes. The orchestrator continuously learns from execution outcomes through the defined reward function that optimises the overall performance of the orchestration decision.

(4) *Heterogeneous Quantum Cloud Environment:* The framework extends the capabilities of QSimPy [26] to models of heterogeneous quantum computing resources (QNodes) with varying capabilities, including qubit counts,

connectivity graphs, gate durations, and error rates. Each node maintains dynamic state information such as queue length and availability, enabling realistic simulation of quantum cloud environments.

(5) *Circuit Transpiler:* This component transforms logical quantum circuits into hardware-specific implementations optimised for the target quantum node's topology and native gate set. The transpilation process is guided by the orchestration decision and affects both execution time and fidelity. We utilised the Qiskit Transpiler[3] with optimisation level 3 as the default transpilation mode.

(6) *QTask Executor:* This component mimics the execution of a quantum task on noisy quantum nodes by analysing the transpiled quantum circuit and estimating execution fidelity and time metrics based on the system model defined in Section 3 and the calibration snapshot data of IBM Quantum systems (with Qiskit Fake Provider[4]).

The framework provides a learning system loop where the DRL agent observes environment states, selects quantum node assignments, receives performance feedback through the reward obtained, and iteratively improves scheduling policies. The integration of realistic transpilation, noise-aware execution modelling, and configurable performance objectives enables the system to learn nuanced scheduling strategies that adapt to varying operational priorities while maintaining practical applicability to practical quantum cloud systems.

This modular architecture supports extensibility for different quantum hardware backends, alternative circuit datasets, and enhanced performance models while maintaining the core orchestration capability through learning-based algorithms with design principles similar to existing works [24, 26].

4.2 Deep Reinforcement Learning Model

Based on the system model and problem formulation in Section 3, we model the quantum task orchestration as a Markov Decision Process (MDP) to enable adaptive policy training through deep reinforcement learning. The MDP is defined by the tuple $(\mathbb{S}, \mathbb{A}, \mathbb{P}, \mathbb{R}, \gamma)$, where $\mathbb{S}$ is the state space, $\mathbb{A}$ the action space, $\mathbb{P}$ the state transition probability, $\mathbb{R}$ the reward function, and $\gamma \in [0, 1]$ the discount factor that balances immediate and future rewards. At each discrete time step t, the agent observes the current state s_t of the environment, selects an action a_t according to a policy $\pi(a_t|s_t)$, and moves to the next state s_{t+1} while receiving a reward r_t. The objective of the agent is to maximise the expected cumulative discounted reward, defined as $\mathbb{V}^{\pi}(s_t) = \mathbb{E}_{\pi}\left[\sum_t \gamma^t r_t\right]$, by learning an optimal policy π. Typically, the policy is parameterised by a neural network and is improved iteratively through training based on observed transitions and rewards.

State Space $\mathbb{S}$: The state s_t at time $t \in \mathbb{T}$ is a concatenation of the feature vector of the current quantum task τ_i and the feature vectors of all available quantum nodes (QNodes) $\mathcal{N}$ in the environment.

$$s_t = \left(\mathbf{f}_t^{\tau_i}, \mathbf{f}_t^{\mathcal{N}}\right) \tag{14}$$

where $\mathbf{f}_t^{\tau_i} \in \mathbb{R}^p$ is the p-dimensional feature vector of the current QTask τ_i, and $\mathbf{f}_t^{\mathcal{N}} \in \mathbb{R}^{m \times o}$ is the concatenated feature matrix of all m QNodes, each with o features. All features are normalised to $[0, 1]$ to ensure stable learning dynamics across heterogeneous scales. Thus, the State space can be defined as:

$$\mathbb{S} = \{s_t | s_t = \left(\mathbf{f}_t^{\tau_i}, \mathbf{f}_t^{\mathcal{N}}\right), \forall t \in \mathbb{T}\} \tag{15}$$

In practical quantum computing environments, frequent calibration is crucial to maintain the stability of quantum nodes and ensure the accuracy of computational results. This process involves regularly adjusting and fine-tuning the

[3]https://quantum.cloud.ibm.com/docs/en/api/qiskit/transpiler
[4]https://quantum.cloud.ibm.com/docs/en/api/qiskit-ibm-runtime/fake-provider

control parameters of qubits, logic gates, and readout mechanisms, leading to dynamic changes in various performance metrics associated with each quantum node. To realistically capture this non-stationary behaviour and to support robust decision-making, we incorporate the complete set of current quantum node metrics into its state representation. As a result, the learning agents can adapt their policies in real time, optimising task orchestration based on the latest calibration data and the current properties of incoming quantum tasks. This adaptive approach enables the system to respond effectively to variations in hardware performance, leading to more reliable and efficient task scheduling.

Action Space $\mathbb{A}$: The action space is discrete and corresponds to the assignment of a suitable QNode for the placement of the current incoming QTask. At each timestep, the agent chooses action $a_t \in \{0, 1, \ldots, m-1\}$, where $m = |\mathcal{N}|$ and $a_t = i$ denotes assigning the task to QNode $n_i \in \mathcal{N}$. Thus,

$$\mathbb{A} = \mathcal{N} \tag{16}$$

Reward Function $\mathbb{R}$: The reward function directly implements the orchestration performance model defined in Equation 9. For each assignment decision to allocate QTask τ_i to QNode n_j at time t, the reward $r_t \in \mathbb{R}$ combines the fidelity score $\mathcal{F}_{i,j}$ and time penalty $\mathcal{T}_{i,j}$ with configurable weight β:

$$r_t = \begin{cases} \mathcal{F}_{i,j} - \beta \cdot \mathcal{T}_{i,j} & \text{if the execution is successful} \\ P_{fail} & \text{otherwise} \end{cases} \tag{17}$$

A failure penalty P_{fail} is applied when task τ_i cannot be executed on the selected node for any reason, providing negative feedback for infeasible assignments to improve the decision of the reinforcement learning policy.

4.3 QFOR: The DRL-based Quantum Fidelity-aware Orchestration Policy

The QFOR technique extends the Proximal Policy Optimisation algorithm [36] to the quantum cloud orchestration problem, with the overall training workflow shown in Algorithm 1. The algorithm operates on the principle of learning an optimal scheduling policy $\pi_\theta : \mathcal{S} \to \Delta(\mathbb{A})$ that maximises long-term cumulative reward while maintaining stable learning dynamics through trust region constraints.

Initially, the algorithm initializes three components: (i) a parameterized policy network π_θ with parameters θ that maps states to action probability distributions, (ii) a value function approximator V_ϕ with parameters ϕ that estimates state values for advantage computation, and (iii) an experience buffer $\mathcal{B}$ for storing trajectory data. Besides, the quantum computing environment instantiation involves configuring heterogeneous quantum nodes to mimic the realistic NISQ hardware parameters, including error rates, gate durations, and qubit connectivity constraints.

The outer training loop iterates over K policy improvement cycles, where each iteration corresponds to one complete policy update using collected experience data. This structure follows the standard PPO algorithm [36], and Ray RLlib [17] of alternating between data collection and policy optimisation phases. The experience buffer is cleared at the beginning of each iteration to ensure on-policy learning. Parallel rollout collection across W workers enables efficient data gathering and improved sample diversity. Each worker operates independently, reducing correlation between consecutive experiences and enhancing the robustness of gradient estimates during policy updates. Environment reset initialises a new episode by sampling the initial QTask τ_0 from the QASM-based [6]circuit dataset $\mathcal{QD}$ and computing the corresponding initial state s_0. The stochastic nature of task arrival ensures diverse training scenarios and prevents overfitting to specific task sequences.

Algorithm 1 QFOR Training Workflow with Proximal Policy Optimisation

Require: QTask dataset $\mathcal{QD}$, QNodes $\mathcal{N}$, hyperparameters Θ
Ensure: Trained policy π_θ

Initialize: Policy network π_θ, value network V_ϕ, experience buffer $\mathcal{B}$
Initialise : Quantum Cloud Environment in QSimPy (QSimPyEnv)
for iteration $k = 1, 2, \ldots, K$ **do**
  $\mathcal{B} \leftarrow \emptyset$ ▷ Clear experience buffer
  for worker $w = 1, 2, \ldots, W$ **do** ▷ Parallel rollouts
    Reset environment: $(s_0, \tau_0) \sim \text{QSimPyEnv}(\mathcal{QD})$
    for step $t = 0, 1, \ldots, T-1$ **do**
      $s_t \leftarrow f_{\text{state}}(\tau_t, \mathcal{N}, t)$ ▷ Normalized features
      $a_t \sim \pi_\theta(\cdot|s_t)$ ▷ Sample action from policy
      $\text{ProcessTask}(\tau_t, n_{a_t})$
      $r_t \leftarrow \text{CaculateReward}()$
      $s_{t+1}, \tau_{t+1} \leftarrow \text{NextTask}()$ ▷ Get next QTask
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ ▷ Store exp.
      if episode terminated or $t = T-1$ **then**
        break
      end if
    end for
  end for
  Policy Update:
  Compute advantages: $\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots$
  Compute returns: $\hat{R}_t = \hat{A}_t + V_\phi(s_t)$
  for epoch $e = 1, 2, \ldots, E$ **do**
    for minibatch $\mathcal{B}_m \subset \mathcal{B}$ **do**
      $r_t(\theta) \leftarrow \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ ▷ Probability ratio
      $L^{\text{CLIP}}(\theta) \leftarrow \mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$
      $L^{\text{VF}}(\phi) \leftarrow \mathbb{E}\left[(V_\phi(s_t) - \hat{R}_t)^2\right]$
      $L^{\text{ENT}}(\theta) \leftarrow \mathbb{E}\left[H(\pi_\theta(\cdot|s_t))\right]$ ▷ Entropy bonus
      $\theta \leftarrow \theta + \alpha_\pi \nabla_\theta (L^{\text{CLIP}}(\theta) + c_1 L^{\text{ENT}}(\theta))$
      $\phi \leftarrow \phi - \alpha_v \nabla_\phi L^{\text{VF}}(\phi)$
    end for
  end for
  $\theta_{\text{old}} \leftarrow \theta$ ▷ Update old policy parameters
end for
return Trained policy π_θ

The inner episode loop processes quantum tasks sequentially until episode termination at the final timestep t or when the task limit is reached. Each step corresponds to scheduling one incoming quantum task to an available quantum node. Action sampling follows the current policy distribution, where a_t represents the selected QNode index. The stochastic policy enables exploration while the learned parameters θ bias the selection toward high-reward actions based on accumulated experience. Each QTask execution involves a circuit transpilation process and performance metrics estimation on the selected quantum node n_{a_t} based on the calibration data of the quantum device. Then, the reward function combines multiple performance indicators as defined (see Equation 17) in the previous section.

State transition involves advancing to the next quantum task in the episode sequence and updating the environment time. The next state s_{t+1} incorporates updated quantum nodes metrics and the new quantum task's characteristics, maintaining the Markov property essential for policy gradient convergence. Experience tuple storage enables subsequent policy optimisation through gradient-based updates. Each tuple (s_t, a_t, r_t, s_{t+1}) provides a complete transition record necessary for advantage estimation and policy gradient computation. Episode termination logic ensures proper boundary handling when task limits are reached or no additional tasks are available. This prevents infinite episodes while maintaining consistent episode lengths for stable learning dynamics.

For the policy optimisation phase, our QFOR technique leverages the standard PPO algorithm [36] and adapts to quantum task orchestration. The advantage computation using Generalised Advantage Estimation (GAE) [35]:

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

where $\delta_t = R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ represents the temporal difference error. GAE balances bias and variance in advantage estimation through the λ parameter.

During the policy update phase, PPO's clipped surrogate objective function is used to ensure bounded policy updates:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

where $r_t(\theta) = \dfrac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$. The value function loss $L^{\text{VF}}(\phi) = \mathbb{E}[(V_\phi(s_t) - \hat{R}_t)^2]$ and entropy bonus $L^{\text{ENT}}(\theta) = \mathbb{E}[H(\pi_\theta(\cdot|s_t))]$ provide additional optimization objectives for stable learning and adequate exploration. The clipped objective function ensures monotonic policy improvement with high probability, while the adaptive reward structure maintains the Markov property essential for convergence and provides scale-invariant rewards across diverse circuit complexities, enabling consistent learning signals.

5 Performance Evaluation

5.1 Environment Setup

We use the QSimPy framework [26] for simulation of quantum cloud environments. We also extended it further to support the modelling of noisy quantum nodes using device calibration data and mimic the practical execution process of a quantum task, which comprises the circuit transpilation to selected QNodes before execution. This approach allows us to emulate heterogeneous quantum cloud computing environments and quantum task execution more comprehensively compared to other existing work [20, 24]. For quantum cloud computation resources, we created a cluster of 5 different quantum nodes ranging from 27 to 127 qubits using the calibration data of IBM devices (using Qiskit FakeBackend instances[5]), including `ibm_auckland`, `ibm_hanoi`, `ibm_kolkata`, `ibm_brisbane`, and `ibm_sherbrooke`.

For quantum tasks, we created different training and evaluation datasets with 16 different quantum benchmark algorithms with qubit numbers ranging from 2 to 27 and initial circuit depths (before transpilation) of 3–30 layers, derived from the MQT Bench dataset [31]. The QTask arrival times were generated following a Poisson distribution, similar to other works [3, 39] to mimic the task arrival at the cloud data centre. We use Gymnasium to wrap the QSimPy-based environment and use Ray RLlib [17] for implementing the reinforcement learning training and using Ray Tune [18] for hyperparameter tuning. All experiments are conducted at the Melbourne Research Cloud computation node with an AMD EPYC 9474F 48-core CPU and 128GB of RAM.

[5]https://quantum.cloud.ibm.com/docs/en/api/qiskit-ibm-runtime/fake-provider

5.2 Performance Study

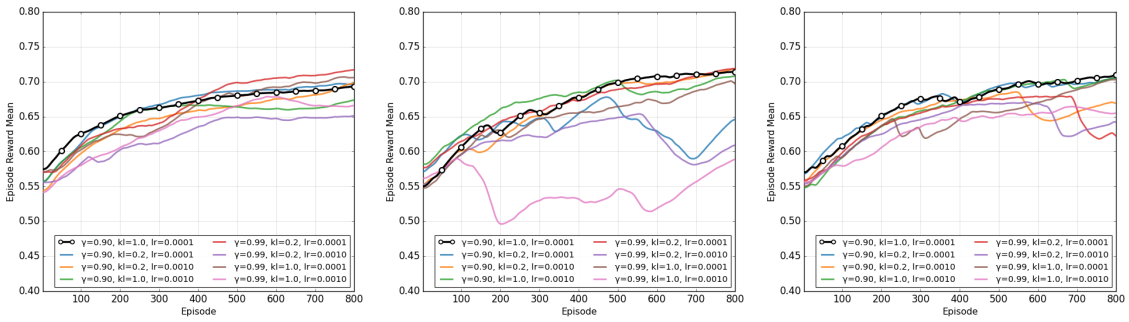
5.2.1 QFOR Policy Training Performance. To thoroughly evaluate the adaptability of QFOR across different operational priorities in quantum cloud environments, with the main priority to optimise the fidelity performance, we trained separate policy instances under different time weights $\beta \in \{0.1, 0.5, 1.0\}$ and different hyperparameters to explore the orchestration trade-off and determine the balance configuration for optimizing the overall orchestration performance. The parameter β determines the balance between optimising fidelity performance, where $\beta = 0.1$ indicates the fidelity-priority mode, while $\beta = 0.5$ and $\beta = 1.0$ promote more balanced trade-offs between fidelity and runtime efficiency. This controlled parameter allows us to characterise the orchestration trade-off and to identify the most suitable configuration for optimising overall performance. We employed Ray Tune [18] for systematic hyperparameter optimisation across all three β configurations, evaluating key parameter combinations to identify optimal settings. The optimal hyperparameter configuration was selected based on convergence stability and final performance across all training modes, with the tuning result shown in Table 2. Other hyperparameters and settings are based on the default configuration of PPO in Ray RLlib [17].

Table 2. QFOR Training Hyperparameters

QFOR Parameters	**Value**
Learning Rate	0.0001
Discount Factor (γ)	0.9
KL Coefficient	1.0
GAE Parameter (λ)	0.95
PPO Clip Parameter	0.3
Entropy Coefficient	0.01
Train Batch Size per Learner	180
Fidelity Reward Weight (α_1, α_2, α_3)	0.8, 0.1, 0.1

Figure 4 shows training convergence across 800 training episodes. The optimal configuration across all configurations demonstrates superior performance with three key advantages: (1) stable reward convergence to approximately 0.70 by training episode 400-500 across all modes, (2) minimal variance indicating robust optimisation dynamics, and (3) consistent performance across different β values, demonstrating hyperparameter robustness. Other configurations exhibited training instability and performance degradation after episode 500-600, particularly in balanced and high-performance modes. The lower discount factor ($\gamma = 0.9$) proves advantageous for quantum orchestration by appropriately balancing immediate scheduling decisions with long-term efficiency in dynamic environments. The average training time for each combination of hyperparameters is approximately 2.577 hrs, 1.735 hrs and 1.857 hrs for $\beta = 0.1$, $\beta = 0.5$, and $\beta = 1.0$, respectively. These training time costs vary based on the computational resources for training and were not affected by the calibration updates, as the deep reinforcement learning agent uses all updated information of available quantum nodes during the training. Therefore, the agent can adaptively improve the model based on the current information of the quantum computing environment.

5.2.2 Execution Fidelity Performance Analysis. To evaluate the effectiveness of the QFOR policy, we conducted a comprehensive performance evaluation across 100 evaluation episodes with 6,000 quantum tasks using an independent test dataset distinct from the training data to ensure an unbiased assessment of the learned policies' generalisation capabilities and practical applicability. We compared QFOR against five representative baseline policies, similar to the evaluation approach of other related works [16, 20, 23, 32]:



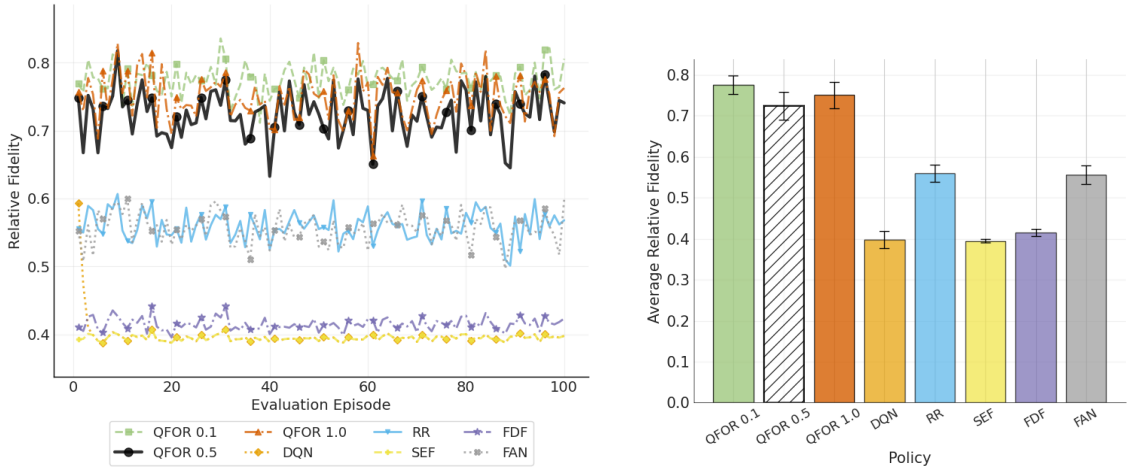
(a) $\beta = 0.1$ (b) $\beta = 0.5$ (c) $\beta = 1.0$

Fig. 4. Training convergence comparison across all configurations. The optimal hyperparameter configuration (black line with markers) achieves more consistent convergence in three different time weight β settings

- *Deep Q-Network (DQN):* Utilises a DQN-based approach, similar to [16, 23] to learn optimal scheduling policies by approximating the action-value function.
- *Round Robin (RR)*: Distributes quantum tasks cyclically across nodes, ensuring fair resource allocation while potentially ignoring node-specific characteristics.
- *Smallest Error First (SEF)*: Assigns QTask to QNode with the smallest average error rates of all gate operations.
- *Fastest Duration First (FDF)*: Assigns QTask to QNode with the fastest average gate duration times.
- *First Available Node (FAN)*: Assigns QTask to the first idle quantum node to minimise waiting time.

The baseline scheduling policies are representative approaches from both classical and quantum resource management literature. Round Robin (RR) and First Available Node (FAN) approaches are widely used classical policies for fair resource allocation, serving as baselines in distributed and quantum settings, which were used as baselines in prior work [16, 23]. Smallest Error First (SEF) and Fastest Duration First (FDF) heuristics reflect the quantum-specific baselines used in recent quantum scheduling study [20, 32], where selection is based on device error rates and execution times. We further considered a Deep Q-Network (DQN) policy, as adopted in recent state-of-the-art DRL-based orchestrators for quantum cloud computing, such as DRLQ [23] and Moirai [16], trained and evaluated under the same quantum cloud simulation for direct comparison with QFOR. In the DQN approach, we use a balanced value of $\beta = 0.5$ for training with a similar configuration in previous work [23]. Figure 5 and Table 3 present the relative fidelity performance comparison across all policies.

QFOR demonstrates substantial fidelity advantages across all operational configurations. The episode-wise analysis in Figure 5a shows QFOR's consistent enhancement throughout the evaluation period. Notably, all three QFOR configurations maintain stable performance in the 0.73-0.78 range, while baselines cluster in the 0.4-0.56 range. As shown in Figure 5b, with error bars denoting standard deviation, QFOR achieves consistently higher relative fidelity performance, with an average fidelity score at 0.776 using $\beta = 0.1$, 0.725 using $\beta = 0.5$, and 0.75 using $\beta = 1.0$. These results represent significant improvements over the best-performing baseline (RR at 0.56), with enhancement margins of 29.5% and 34%, respectively. While DQN learns scheduling policies by approximating action-value functions via deep neural networks, our results show that QFOR consistently outperforms DQN in terms of fidelity and overall orchestration balance (see Table 3). Notably, SEF policy, which greedily selects the QNode with the smallest average



(a) Execution fidelity in all evaluation episodes

(b) Average execution fidelity of all policies

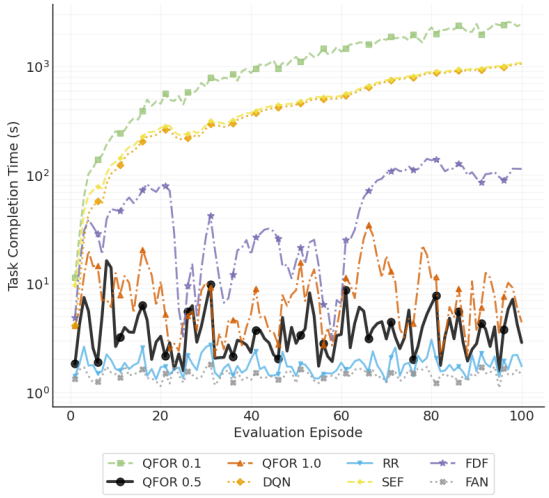
Fig. 5. Relative execution fidelity performance comparison across all policies over 100 evaluation episodes.

error rate regardless of the scheduled quantum circuit structure, fails to maintain a high fidelity performance score, achieving a mean performance score of 0.395, further justifying the effectiveness of our complexity bonus and balanced reward design. This highlights that final execution fidelity depends not only on the average error rate of the QNode but also critically on the transpilation process, including the mapping of the QTask to the specific qubit topology of the QNode. These results indicate the key limitations of conventional scheduling approaches in quantum cloud environments and demonstrate robust policy learning that generalises effectively to unseen quantum tasks and dynamic device conditions. The consistent high performance across different β values demonstrates the ability of QFOR to learn context-aware scheduling policies that adapt optimisation priorities while maintaining overall effectiveness.

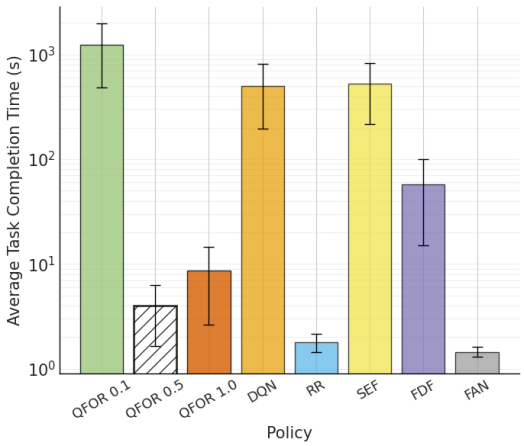
Table 3. Detailed performance comparison of all policies, regarding average relative fidelity score, execution time, and total completion time (± standard deviation) over 100 evaluation episodes

Policy	Average Fidelity Score	Average Execution Time (s)	Average Completion Time (s)
QFOR ($\beta = 0.1$)	0.776 ± 0.023	1.725 ± 0.198	1229.711 ± 746.596
QFOR ($\beta = 0.5$)	0.725 ± 0.034	1.104 ± 0.131	3.967 ± 2.336
QFOR ($\beta = 1.0$)	0.750 ± 0.033	1.412 ± 0.161	8.580 ± 5.946
DQN	0.398 ± 0.022	1.224 ± 0.125	501.228 ± 306.203
RR	0.560 ± 0.021	1.523 ± 0.174	1.784 ± 0.357
SEF	0.395 ± 0.004	1.220 ± 0.123	521.675 ± 306.631
FDF	0.415 ± 0.009	1.039 ± 0.114	57.502 ± 42.483
FAN	0.556 ± 0.022	1.444 ± 0.157	1.448 ± 0.159

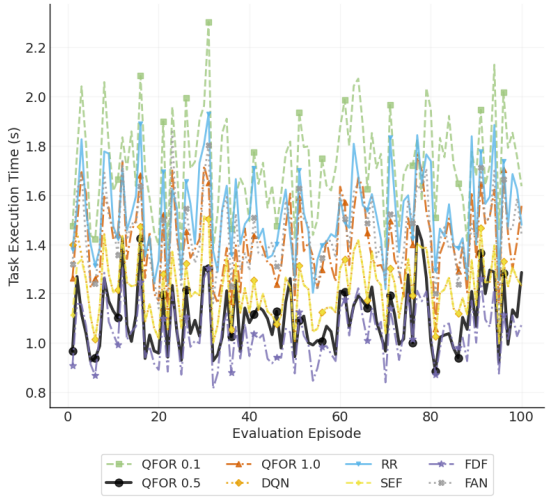
5.2.3 Execution Fidelity-Time Trade-off Analysis. As the main priority of the orchestration is optimizing the fidelity performance, with the secondary consideration being to balance the execution time required, we conducted a comprehensive analysis of total completion time and quantum execution time of all QTasks in the evaluation across all policies



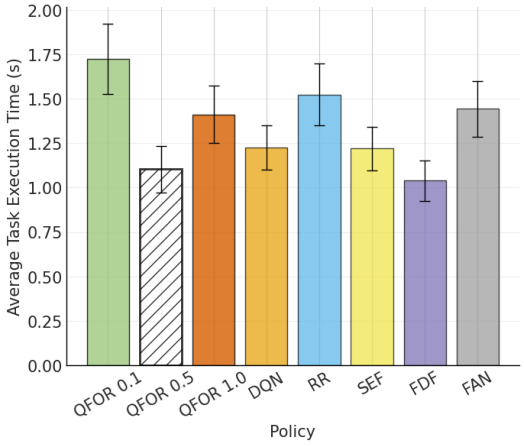
(a) Task completion time across evaluation episodes



(b) Average task completion time comparison



(c) Task execution time across evaluation episodes



(d) Average task execution time comparison

Fig. 6. Task completion and quantum execution time analysis of all policies over 100 evaluation episodes.

to find the optimal time weight to achieve this orchestration goal and explore the trade-off between fidelity and time in the decision. Figure 6 and Table 3 present a detailed timing analysis across all policies, demonstrating both average performance and episode-wise progression over 100 evaluation episodes.

The results indicate that greedy policies focusing exclusively on gate error rates (SEF) or gate duration time (FDF) lead to substantial waiting times, resulting in significantly longer total completion times compared to other approaches. In contrast, policies such as First Available Node (FAN) and Round Robin (RR) effectively minimise waiting times by distributing tasks more evenly, thereby reducing overall completion time. As shown in Figure 5, the fidelity-priority

setting of QFOR ($\beta = 0.1$) achieves the highest fidelity among all policies. However, as highlighted in Figure 6 and Table 3, this comes at the expense of extremely long overall task completion times, illustrating the downside of greedy fidelity-centric scheduling, leading to the potential starvation of high-quality QNodes and growing queue imbalances. This underscores the need for a balanced multi-objective design.

As quantum execution time cost is critical and task queueing before execution can be handled by a classical controller, it is more valuable and cost-efficient to optimise the quantum execution time. Figures 6c and 6d show the actual quantum execution time of all policies. The result shows that the balanced time weight $\beta = 0.5$ achieves a reasonable average execution time, which is slightly higher than the execution time of the greedy policy (FDF), while maintaining the fidelity score consistently higher than all of the other baselines, up to approximately 84% higher relative fidelity performance score compared to the SEF baseline. This result suggests the optimal and balanced configurations of the time weight that encourage the QFOR policy to achieve high fidelity while maintaining the balance of fidelity-time tradeoff. Furthermore, minimising the execution time also inherently optimises the monetary cost of using quantum resources, which is essential in the current landscape of quantum cloud computing environments [22].

The results demonstrate that adaptive orchestration fundamentally balances the execution fidelity-time trade-off. While traditional approaches force a binary choice between speed and quality, QFOR policy identifies strategies that balance both objectives. This analysis establishes three key insights for quantum cloud orchestration: (1) Fidelity should be the key consideration in the current NISQ era as error sensitivity makes quality optimization essential as fast but inaccurate operations are ultimately ineffective, (2) Our adaptive, configurable policy enable flexible resource management optimization strategies which can be further extended, and (3) DRL-based is a potential approach that discovers non-obvious scheduling patterns that outperform conventional heuristics.

6 Conclusion and Future Directions

This paper presents QFOR, a novel fidelity-aware deep reinforcement learning framework for quantum task orchestration in heterogeneous cloud-based environments with NISQ computation resources. We developed a holistic orchestration technique specifically designed for optimising overall fidelity performance, which is critical for quantum task execution. Our systematic approach integrates comprehensive quantum task execution and fidelity performance estimators based on device calibration data, enabling rigorous emulation of noisy quantum hardware behaviour. The configurable orchestration objectives successfully balance execution fidelity and time across different operational priorities, with extensive evaluation demonstrating 29.5-84% fidelity performance score improvements over other existing deep reinforcement learning and heuristic baseline methods. This work establishes a foundation and facilitates learning-driven research on quantum resource management in hybrid quantum-HPC systems, which require substantial effort to develop robust, adaptive frameworks and techniques to keep pace with advances in quantum hardware.

While QFOR demonstrates potential in quantum cloud orchestration, we recognised several limitations that present opportunities for future research. First, dynamic device calibration data can be used to mimic continuously updated device characteristics and to adapt scheduling policies in real time. Second, as the number of physical qubits in the quantum node increases, training the orchestration policy may require more time and computational resources. Therefore, we plan to investigate QFOR performance and retraining overhead on large-scale quantum cloud infrastructures with dozens of quantum devices, each with a larger number of qubits. We also plan to incorporate more rigorous approaches to multi-objective optimisation in future work. Furthermore, circuit knitting [27], which distributes large quantum tasks across distributed quantum systems, can be considered to improve suitability for NISQ devices. Additionally, the centralised policy-learning architecture may become a bottleneck in distributed quantum computing environments,

motivating future exploration of distributed reinforcement-learning architectures [11, 30], in which multiple agents coordinate scheduling decisions across geographically distributed quantum resources. These future directions will be essential for realising the full potential and optimising resource management in quantum cloud computing as the technology matures toward practical applications.

Acknowledgments

Hoa Nguyen acknowledges the support from the Science and Technology Scholarship Program for Overseas Study for Doctoral Degrees, Vingroup, Vietnam and studentship from Melbourne qCLOUDS Lab.

References

[1] Thomas Beck, Alessandro Baroni, Ryan Bennink, Gilles Buchs, Eduardo Antonio Coello Pérez, Markus Eisenbach, Rafael Ferreira Da Silva, Muralikrishnan Gopalakrishnan Meena, Kalyan Gottiparthi, Peter Groszkowski, Travis S. Humble, Ryan Landfield, Ketan Maheshwari, Sarp Oral, Michael A. Sandoval, Amir Shehata, In-Saeng Suh, and Christopher Zimmer. 2024. Integrating quantum computing resources into scientific HPC ecosystems. *Future Generation Computer Systems* 161 (Dec. 2024), 11–25. doi:10.1016/j.future.2024.06.058

[2] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (Sept. 2017), 195–202. doi:10.1038/nature23474

[3] Dario Bruneo. 2014. A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems* 25, 3 (March 2014), 560–569. doi:10.1109/tpds.2013.67

[4] Jwo-Sy Chen, Erik Nielsen, Matthew Ebert, Volkan Inlek, Kenneth Wright, Vandiver Chaplin, Andrii Maksymov, Eduardo Páez, Amrit Poudel, Peter Maunz, and John Gamble. 2023. Benchmarking a trapped-ion quantum computer with 29 algorithmic qubits. http://arxiv.org/abs/2308.05071 arXiv:2308.05071 [quant-ph].

[5] Zheyi Chen, Jia Hu, Geyong Min, Chunbo Luo, and Tarek El-Ghazawi. 2022. Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement Learning. *IEEE Transactions on Parallel and Distributed Systems* 33, 8 (Aug. 2022), 1911–1923. doi:10.1109/TPDS.2021.3132422

[6] Andrew Cross, Ali Javadi-Abhari, Thomas Alexander, Niel De Beaudrap, Lev S. Bishop, Steven Heidel, Colm A. Ryan, Prasahnt Sivarajah, John Smolin, Jay M. Gambetta, and Blake R. Johnson. 2022. OpenQASM 3: A Broader and Deeper Quantum Assembly Language. *ACM Transactions on Quantum Computing* 3, 3 (Sept. 2022), 1–50. doi:10.1145/3505636

[7] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. 2019. Validating quantum computers using randomized model circuits. *Physical Review A* 100, 3 (Sept. 2019), 032328. doi:10.1103/PhysRevA.100.032328

[8] Aniello Esposito, Jessica R. Jones, Sebastien Cabaniols, and David Brayford. 2023. A Hybrid Classical-Quantum HPC Workload. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Bellevue, WA, USA, 117–121. doi:10.1109/QCE57702.2023.10194

[9] Yuping Fan, Boyang Li, Dustin Favorite, Naunidh Singh, Taylor Childers, Paul Rich, William Allcock, Michael E. Papka, and Zhiling Lan. 2022. DRAS: Deep Reinforcement Learning for Cluster Scheduling in High Performance Computing. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (Dec. 2022), 4903–4917. doi:10.1109/TPDS.2022.3205325

[10] Emmanouil Giortamis, Francisco Romão, Nathaniel Tornow, Dmitry Lugovoy, and Pramod Bhatotia. 2024. Orchestrating Quantum Cloud Environments with Qonductor. http://arxiv.org/abs/2408.04312 arXiv:2408.04312 [quant-ph].

[11] Mohammad Goudarzi, Marimuthu S Palaniswami, and Rajkumar Buyya. 2021. A Distributed Deep Reinforcement Learning Technique for Application Placement in Edge and Fog Computing Environments. *IEEE Transactions on Mobile Computing* (2021), 1–1. doi:10.1109/TMC.2021.3123165

[12] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2017. Rainbow: Combining Improvements in Deep Reinforcement Learning. http://arxiv.org/abs/1710.02298 arXiv:1710.02298 [cs].

[13] Mohammad Kordzanganeh, Markus Buchberger, Basil Kyriacou, Maxim Povolotskii, Wilhelm Fischer, Andrii Kurkin, Wilfrid Somogyi, Asel Sagingalieva, Markus Pflitsch, and Alexey Melnikov. 2023. Benchmarking Simulated and Physical Quantum Processing Units Using Quantum and Hybrid Algorithms. *Advanced Quantum Technologies* 6, 8 (Aug. 2023), 2300043. doi:10.1002/qute.202300043

[14] Arthur Kurlej, Sam Alterman, and Kevin Obenland. 2024. Performance of algorithms for emerging ion-trap quantum hardware. *Future Generation Computer Systems* 160 (Nov. 2024), 654–665. doi:10.1016/j.future.2024.06.005

[15] Jinyang Li, Yuhong Song, Yipei Liu, Jianli Pan, Lei Yang, Travis Humble, and Weiwen Jiang. 2025. QuSplit: Achieving Both High Fidelity and Throughput via Job Splitting on Noisy Quantum Computers. doi:10.48550/arXiv.2501.12492 arXiv:2501.12492 [quant-ph].

[16] Tingting Li and Ziming Zhao. 2024. Moirai: optimizing quantum serverless function orchestration via device allocation and circuit deployment. In *2024 IEEE International Conference on Web Services (ICWS)*. 707–717. doi:10.1109/ICWS62655.2024.00090

[17] Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E Gonzalez, Michael I Jordan, and Ion Stoica. 2018. RLlib: abstractions for distributed reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR, Stockholm, Sweden.

[18] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. 2018. Tune: a research platform for distributed model selection and training. In *ICML 2018 AutoML Workshop*. PMLR, Stockholm, Sweden. doi:10.48550/arXiv.1807.05118 arXiv:1807.05118 [cs].

[19] Phillip C. Lotshaw, Thien Nguyen, Anthony Santana, Alexander McCaskey, Rebekah Herrman, James Ostrowski, George Siopsis, and Travis S. Humble. 2022. Scaling quantum approximate optimization on near-term hardware. *Scientific Reports* 12, 1 (July 2022), 12388. doi:10.1038/s41598-022-14767-w

[20] Waylon Luo, Jiapeng Zhao, Tong Zhan, and Qiang Guan. 2025. Adaptive Job Scheduling in Quantum Clouds Using Reinforcement Learning. doi:10.48550/arXiv.2506.10889 arXiv:2506.10889 [cs].

[21] Nikolaj Moll, Panagiotis Barkoutsos, Lev S Bishop, Jerry M Chow, Andrew Cross, Daniel J Egger, Stefan Filipp, Andreas Fuhrer, Jay M Gambetta, Marc Ganzhorn, Abhinav Kandala, Antonio Mezzacapo, Peter Müller, Walter Riess, Gian Salis, John Smolin, Ivano Tavernelli, and Kristan Temme. 2018. Quantum optimization using variational algorithms on near-term quantum devices. *Quantum Science and Technology* 3, 3 (July 2018), 030503. doi:10.1088/2058-9565/aab822

[22] Hoa T. Nguyen, Prabhakar Krishnan, Dilip Krishnaswamy, Muhammad Usman, and Rajkumar Buyya. 2024. Quantum Cloud Computing: A Review, Open Problems, and Future Directions. http://arxiv.org/abs/2404.11420 arXiv:2404.11420 [cs].

[23] Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya. 2024. DRLQ: A Deep Reinforcement Learning-based Task Placement for Quantum Cloud Computing. In *2024 IEEE 17th International Conference on Cloud Computing (CLOUD)*. IEEE, Shenzhen, China, 475–481. doi:10.1109/CLOUD62652.2024.00060

[24] Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya. 2024. iQuantum: a toolkit for modeling and simulation of quantum computing environments. *Software: Practice and Experience* 54, 6 (June 2024), 1141–1171. doi:10.1002/spe.3331

[25] Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya. 2024. QFaaS: A Serverless Function-as-a-Service framework for Quantum computing. *Future Generation Computer Systems* 154 (May 2024), 281–300. doi:10.1016/j.future.2024.01.018

[26] Hoa T. Nguyen, Muhammad Usman, and Rajkumar Buyya. 2025. QSimPy: a learning-centric simulation framework for quantum cloud resource management. In *Quantum Computing*. Morgan Kaufmann, 165–183. doi:10.1016/B978-0-443-29096-1.00012-X

[27] Christophe Piveteau and David Sutter. 2024. Circuit Knitting With Classical Communication. *IEEE Transactions on Information Theory* 70, 4 (April 2024), 2734–2745. doi:10.1109/TIT.2023.3310797

[28] Christopher Portmann and Renato Renner. 2022. Security in quantum cryptography. *Reviews of Modern Physics* 94, 2 (June 2022). doi:10.1103/revmodphys.94.025008

[29] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. doi:10.22331/q-2018-08-06-79

[30] Xiaoyu Qiu, Weikun Zhang, Wuhui Chen, and Zibin Zheng. 2021. Distributed and Collective Deep Reinforcement Learning for Computation Offloading: A Practical Perspective. *IEEE Transactions on Parallel and Distributed Systems* 32, 5 (May 2021), 1085–1101. doi:10.1109/TPDS.2020.3042599

[31] Nils Quetschlich, Lukas Burgholzer, and Robert Wille. 2023. MQT Bench: Benchmarking Software and Design Automation Tools for Quantum Computing. *Quantum* 7 (July 2023), 1062. doi:10.22331/q-2023-07-20-1062 arXiv:2204.13719 [quant-ph].

[32] Gokul Subramanian Ravi, Kaitlin N. Smith, Prakash Murali, and Frederic T. Chong. 2021. Adaptive job and resource management for the growing quantum cloud. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Broomfield, CO, USA, 301–312. doi:10.1109/QCE52317.2021.00047

[33] Salonik Resch and Ulya R. Karpuzcu. 2022. Benchmarking Quantum Computers and the Impact of Quantum Noise. *Comput. Surveys* 54, 7 (2022). doi:10.1145/3464420 arXiv: 1912.00546.

[34] Nishant Saurabh, Shantenu Jha, and Andre Luckow. 2023. A Conceptual Architecture for a Quantum-HPC Middleware. In *2023 IEEE International Conference on Quantum Software (QSW)*. IEEE, Chicago, IL, USA, 116–127. doi:10.1109/QSW59989.2023.00023

[35] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-dimensional continuous control using generalized advantage estimation. doi:10.48550/arXiv.1506.02438 arXiv:1506.02438 [cs].

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. http://arxiv.org/abs/1707.06347 arXiv:1707.06347 [cs].

[37] Andrew Wack, Hanhee Paik, Ali Javadi-Abhari, Petar Jurcevic, Ismael Faro, Jay M. Gambetta, and Blake R. Johnson. 2021. Quality, Speed, and Scale: three key attributes to measure the performance of near-term quantum computers. http://arxiv.org/abs/2110.14108 arXiv:2110.14108.

[38] Deliang Yi, Xin Zhou, Yonggang Wen, and Rui Tan. 2020. Efficient Compute-Intensive Job Allocation in Data Centers via Deep Reinforcement Learning. *IEEE Transactions on Parallel and Distributed Systems* 31, 6 (June 2020), 1474–1485. doi:10.1109/TPDS.2020.2968427

[39] Min Sang Yoon, Ahmed E. Kamal, and Zhengyuan Zhu. 2017. Adaptive data center activation with user request prediction. *Computer Networks* 122 (July 2017), 191–204. doi:10.1016/j.comnet.2017.04.047

[40] Shaojun Zhang, Chen Wang, and Albert Y. Zomaya. 2022. Robustness Analysis and Enhancement of Deep Reinforcement Learning-based Schedulers. *IEEE Transactions on Parallel and Distributed Systems* (2022), 1–12. doi:10.1109/TPDS.2022.3218649

[41] Maximillian Zinner, Florian Dahlhausen, Philip Boehme, Jan Ehlers, Linn Bieske, and Leonard Fehring. 2021. Quantum computing's potential for drug discovery: Early stage industry dynamics. *Drug Discovery Today* 26, 7 (July 2021), 1680–1688. doi:10.1016/j.drudis.2021.06.003