Qoncord: A Multi-Device Job Scheduling Framework for Variational Quantum Algorithms

Meng Wang
ECE Department
The University of British Columbia
Vancouver, Canada
mengwang@ece.ubc.ca

Poulami Das
ECE Department
The University of Texas at Austin
Austin, USA
poulami.das@utexas.edu

Prashant J. Nair
ECE Department
The University of British Columbia
Vancouver, Canada
prashantnair@ece.ubc.ca

Abstract—Quantum computers face challenges due to limited resources, particularly in cloud environments. Despite these obstacles, Variational Quantum Algorithms (VQAs) are considered promising applications for present-day Noisy Intermediate-Scale Quantum (NISQ) systems. VQAs require multiple optimization iterations to converge on a globally optimal solution. Moreover, these optimizations, known as restarts, need to be repeated from different points to mitigate the impact of noise. Unfortunately, the job scheduling policies for each VQA task in the cloud are heavily unoptimized. Notably, each VQA execution instance is typically scheduled on a single NISQ device. Given the variety of devices in the cloud, users often prefer higher-fidelity devices to ensure higher-quality solutions. However, this preference leads to increased queueing delays and unbalanced resource utilization.

We propose *Qoncord*, an automated job scheduling framework to address these cloud-centric challenges for VQAs. Qoncord leverages the insight that not all training iterations and restarts are equal, Qoncord strategically divides the training process into exploratory and fine-tuning phases. Early exploratory iterations, more resilient to noise, are executed on less busy machines, while fine-tuning occurs on high-fidelity machines. This adaptive approach mitigates the impact of noise, optimizes resource usage, and reduces queuing delays in cloud environments. Qoncord also significantly reduces execution time and minimizes restart overheads by eliminating low-performance iterations. Thus, Qoncord offers similar solutions 17.4× faster. It also provides 13.3% better solutions for the same time budget as the baseline.

Keywords—Quantum Computing, Cloud Environment, Variational Quantum Algorithm

I. INTRODUCTION

Quantum computers hold the promise of transformative computational capabilities [1]–[4]. While quantum computers with a few hundred qubits are accessible through cloud services from IBM, Amazon, and Microsoft [5]–[8], their potential is hindered by the inherent hardware noise and scarcity of resources. Near-term *Noisy Intermediate-Scale Quantum (NISQ)* systems [9] lack the redundancies necessary for fault tolerance [10]. Nevertheless, NISQ devices promise to accelerate many crucial domain-specific applications using *Variational Quantum Algorithms (VQAs)* [4], [11], [12]. These algorithms map the problems onto parametric quantum circuits and iteratively train circuit parameters with a classical optimizer. However, the performance of VQAs is significantly constrained by noise and prolonged device access times.

Program execution on NISQ devices significantly affects the training process of VQAs. For instance, noisy executions can necessitate increased iterations due to a lack of gradients or, in the worst case, not converging to the optimal parameters. The optimization process often gets trapped in local optima, prompting the need for multiple random *restarts* [13], [14]. In these restarts, end-to-end optimization is repeated from a different starting point, and the best result is ultimately selected. Consequently, hardware noise causes the total number of circuit executions required for VQAs to surge compared to an idealized noise-free setting [15]–[17].

These problems are exacerbated by (1) the limited availability of quantum computers in the cloud and (2) device-level variations leading to divergent error profiles. Globally, nearly a dozen *quantum cloud services* provide access to fewer than fifty quantum systems [5]–[7], [18]–[22]. These scarce resources are shared among thousands of users, resulting in significant queueing delays. This bottleneck is pronounced for VQAs, which execute circuits sequentially, with hundreds of computational jobs often awaiting execution on a single device [23]. Additionally, quantum computers vary in size and error rates. For instance, IBM Quantum systems range from 27 to 127 qubits, with 2-qubit gate error rates varying by 7× (from 0.3% to 2.1%) [5]. This leads to unbalanced loads as users and cloud providers naturally select the highest fidelity devices, further complicating resource allocation.

Presently, three approaches exist to mitigate cloud latencies. *Firstly*, IBM's Runtime allows *dedicated* device access to users [24]. Users can specify a device, or the cloud provider defaults to the least busy device based on the load [25]. *Secondly*, Ravi et al. suggest prioritizing a high-fidelity device with a low load and opting for a low-fidelity device otherwise [23]. However, this approach may select low-fidelity devices during periods of high load, potentially compromising solution quality and training performance. *Thirdly*, Stein et al. propose a framework mainly applicable to a class of VQAs for chemistry applications [26] that require many circuit executions per iteration. The framework distributes the
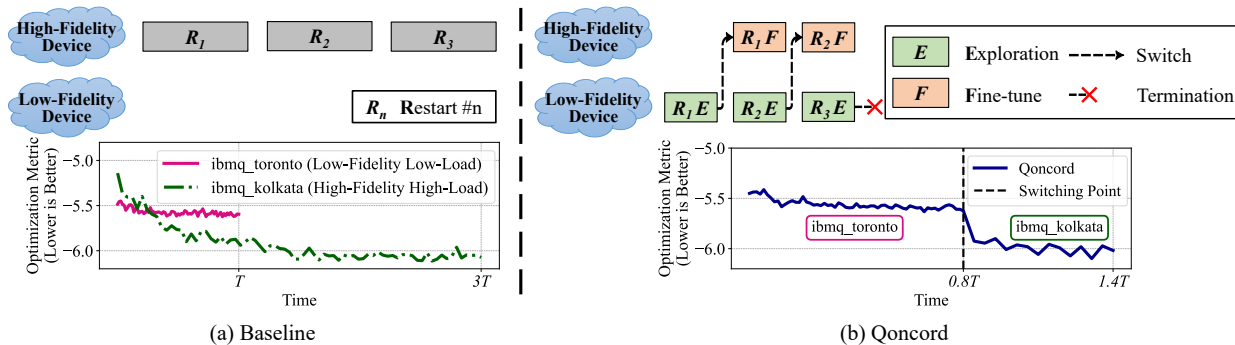
Fig. 1. In the baseline approach (Figure (a)), all VQA iterations for each restart are executed on a single device. Despite queueing delays, ibmq_kolkata outperforms ibmq_toronto by 159%, even with 3× more pending jobs (with longer actual wait times). On the other hand, Qoncord (Figure (b)) starts exploratory iterations on the low-load (LL) device, ibmq_toronto, before shifting to the high-fidelity ibmq_kolkata for fine-tuning. This dynamic scheduling leads to 2.14× faster performance. Additionally, Qoncord efficiently evaluates restart quality on the low-load device, promptly terminating low-quality restarts.

executions across multiple devices (with high and low loads) to accelerate gradient estimation in each iteration. However, all these scheduling policies use the same device(s) *throughout the training*, as shown in Figure 1(a). Thus, their performance is limited by either the noise of the lowest-fidelity device or the wait times of the highest-load device.

Enhancing the performance of VQAs is challenging due to conflicting objectives; devices with *Low-Fidelity* typically exhibit *Low-Load*, whereas *High-Fidelity* devices often experience *High-Load*. Figure 1(a) shows the training steps of a VQA on a noisy simulator using the error model of two 27-qubit IBMQ devices[1]. We observe that utilizing Low-Fidelity Low-Load devices reduces execution time but compromises accuracy. The ibmq_kolkata device, with higher fidelity than the ibmq_toronto device, achieves a 159% improvement in optimization performance; however, it also encounters prolonged queueing delays, which results in extended execution times. This paper aims to improve throughput (using Low-Fidelity **Low-Load** devices) without compromising solution quality (using **High-Fidelity** High-Load devices).

This paper introduces *Qoncord*, an automated job scheduling framework tailored to overcome the above challenges for quantum clouds. Qoncord leverages two insights:

1. Not all Training Steps are Equal: Qoncord is built on the insight that *not all VQA training steps contribute equally* to the optimization process. It leverages this understanding to recommend that different training steps need not execute on the same quantum device. Thus, Qoncord divides the training process into distinct stages. During the initial *exploration* stage, the optimization routine explores a broad parameter region to identify potential areas containing the optima. The subsequent *fine-tuning* stage involves precisely adjusting the parameter settings to converge to the optimal solution.

Our results, shown in Section IV-B, reveal that the exploration stage is more resilient to noise than the fine-tuning stage. Thus, Qoncord directs the exploration to a Low-Fidelity Low-Load device and dynamically switches to a High-Fidelity

High-Load device for performance-critical fine-tuning. As shown in Figure 1(b), this transition occurs if the optimization metric remains unchanged over a specified number (*threshold*) of iterations on the Low-Fidelity Low-Load device. Qoncord achieves fidelity comparable to High-Fidelity-only access in less time by using this scheduling approach.

2. Not all Restarts are Equal: Qoncord leverages the insight that *not all restarts are equal*, and therefore, not all of them have to go through the end-to-end optimization routine. As illustrated in Figure 1(b), Qoncord quickly evaluates the early exploration steps of each restart on the Low-Fidelity Low-Load device, terminates poor candidates, and fine-tunes the remaining candidates on the High-Fidelity High-Load device. By doing so, the effective throughput of the VQA execution is improved even further. Although Figure 1(b) only illustrates the splitting of the execution into two phases on two devices, Qoncord can be generalized to split the execution into a larger number of phases and then run them on a fleet of NISQ devices with varying fidelities.

To that end, this paper makes four key contributions:

1) We quantitatively demonstrate that high-fidelity quantum devices experience higher loads than other devices.
2) We show that running all VQA iterations on the same device suffers from poor accuracy or high execution time.
3) We propose Qoncord, an automated software framework that splits VQA training into phases and adaptively schedules them across multiple devices.
4) Qoncord reduces the overheads of restarts during VQA training by adaptively terminating poor candidates.
5) Qoncord provides incentives for device vendors to release non-optimal devices by demonstrating their effective utilization. Thus, it enables cloud service providers to maximize resource utility and reduce queuing delays.

Our evaluations using the error models of IBMQ quantum devices show that Qoncord offers (1) similar solution quality for VQAs 17.4× faster and (2) provides 13.3% times better solution quality within the same training time.

---

[1]It is noteworthy that even with dedicated access to ibmq_kolkata, users are constrained by access times, especially if numerous users request access.

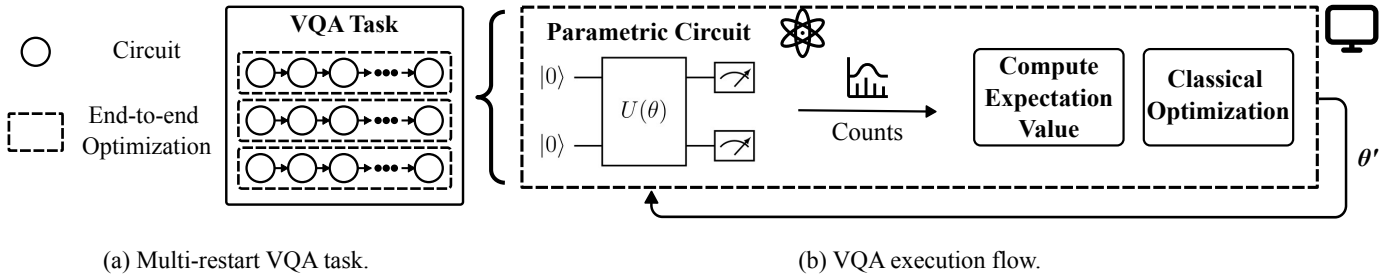(a) Multi-restart VQA task.　　　　　　　　　　　　　　　(b) VQA execution flow.

Fig. 2. (a) Sample VQA task with three random restarts. (b) VQA executes a parametric circuit whose parameters are tuned over hundreds of iterations by a classical optimization routine. The expectation value from the output distribution is used to update the circuit parameters for the next iteration.

II. Background

A. Variational Quantum Algorithms (VQAs)

Variational Quantum Algorithms (VQAs) promise to accelerate diverse applications in healthcare [27], combinatorial optimizations [4], quantum chemistry [11], [28], and machine learning [29] using NISQ devices. These algorithms utilize a parameterized quantum circuit, as illustrated in Figure 2. The circuit's gate parameters undergo training via a classical optimizer, optimizing an objective function corresponding to the specific problem over numerous iterations.

The expectation value is computed from the output distribution in each iteration. This metric guides the adjustment of circuit parameters for the subsequent iteration. The training process continues until the optimal parameters are identified. At this convergence point, the circuit's output corresponds to the optimal solution for the given problem.

B. Impact of Quantum Hardware Errors on VQAs

Noisy quantum hardware hampers VQA performance by extending training time and hindering convergence due to lack of gradients [30]. Instead of arriving at any global optima, the optimization process can get trapped in a local optima. To overcome this, VQAs employ a *multi-restart* strategy that executes independent optimization iterations with different initial parameters, as shown in Figure 2. This increases the chances of discovering the global optimum with the cost of additional circuit executions. The number of restarts, typically a few hundred, depends on the circuit complexity [31].

C. Quantum Computing in the Cloud

Operating quantum computers requires specialized infrastructures and high-precision control electronics. These requirements incur substantial deployment and operational costs. To address these challenges, *Quantum Cloud Services (QCS)* provides a vital interface that shields users from the complexities of quantum hardware and facilitates seamless remote access. Major device manufacturers, including IBM, Rigetti, and IonQ, offer direct access to their quantum processing units (QPUs) through cloud services. Cloud service providers like Amazon Web Services (AWS) and Microsoft Azure enable access to third-party QPUs.

D. Demand vs. Supply Gap In Quantum Clouds

Enterprises such as Boeing, JSR, Exxon, Mitsubishi Chemical, Daimler, CERN, and others are exploring quantum applications through partnerships with QCS providers [32]–[44]. IBM Quantum alone collaborates with 210+ organizations, spanning Fortune 500 companies, universities, and startups [5].

Despite this growth, the demand for quantum resources significantly outpaces availability. Globally, only a dozen QCS providers offer access to less than 50 machines [5]–[7], [18]–[22]. This considerable gap between demand and supply cannot be bridged solely by building more quantum computers due to their high operational and maintenance expenses. Consequently, QCS providers must intelligently schedule their limited quantum resources among various users.

E. Job Scheduling Policies

Access to quantum cloud follows two models: *Shared* and *Runtime*. Both models employ fair-share scheduling, ensuring equitable distribution of computation time among users.

1. Shared Access Model: In this model, users submit individual jobs, which are then queued for execution on the quantum device. The fair-share scheduling algorithm determines a job's position in the queue. It considers factors such as the number of requests, the requested computation time, and the user's past usage to allocate resources.

2. Runtime Access Model: The runtime model [24] offers a more dynamic interaction with quantum resources. Users establish a continuous session upon gaining access to a quantum device. Users can submit multiple jobs once a session is established without rejoining the queue. Jobs within an active session receive higher priority for execution. This benefits VQAs by reducing latency between iterations. Although the runtime model facilitates a more efficient submission process during an active session, it does not alleviate the underlying queuing challenge. This is because the demand for quantum devices still exceeds supply. Thus, jobs from other users still need to wait while the runtime model prioritizes a user.

These access models overcome the limitations of a single machine chosen to execute all program iterations. Their scheduling algorithm does not consider inter-machine characteristics. Some QCS providers are pursuing intelligent models to counter this. For instance, IBM's recent video for their runtime model envisions a future where the jobs are distributed

over multiple machines. However, to our knowledge, no such scheduling algorithm is currently available [45].

III. MOTIVATION

A. Balancing Loads and Fidelity in Quantum Clouds

QCS encounters extreme load imbalances. These are primarily driven by variations in device-level error characteristics within diverse quantum computers. Typically, high-fidelity machines experience heavier loads due to the natural inclination to run applications on these devices, creating an imbalance with respect to low-fidelity systems. For instance, Table I shows the average wait times for different machines from Rigetti and IonQ. The number of algorithmic qubits (#AQ) is a system-level metric used to benchmark the performance of IonQ machines, and a higher value is desirable [46]–[48]. This value is unavailable for providers like Rigetti. Thus, we also compare the average 2-qubit gate fidelities.

TABLE I
FIDELITY AND WAIT TIMES COMPARISON OF DEVICES

Provider	Device	Gate Fidelity (%)	#AQ	Wait Time
Rigetti [49]	Aspen-M-3	94.6	-	4 hours
IonQ [50]	Harmony	97.1	25	1.9 days
	Aria	98.9	25	10.7 days
	Forte	99.4	29	7 days

Notably, the wait times for noisier Rigetti machines are $10.9\times$ to $61.3\times$ lower than those for high-fidelity IonQ machines. There is variability even within the same provider (IonQ). Aria and Forte, with higher fidelity, exhibit $3.7\times$ to $5.6\times$ longer wait times than Harmony. This unique trade-off presents QCS providers with a choice: either prioritize faster time to solution on low-fidelity devices at the expense of solution quality or endure prolonged wait times for high-fidelity machines to achieve better solution quality.

B. Variance in Capabilities of Quantum Systems

Quantum systems exhibit a trade-off between fidelity and program execution time due to three fundamental reasons:

1) Different qubit device technologies (systems from multiple vendors): Multiple qubit technologies, including superconducting qubits, trapped ions, and neutral atoms, are being explored to build large-scale quantum computers. Each device technology is associated with its unique execution and error characteristics. They also mature at different rates due to the unique challenges associated with scaling each qubit technology. As shown in Table I, IonQ's trapped ion devices offer higher fidelity but are more than $1000\times$ slower (measured as time per gate) than Rigetti's superconducting qubits. Thus, program execution is orders of magnitude slower on IonQ systems compared to Rigetti, even if cloud access latencies are considered to be negligible (assuming user reserves the machine). The problem worsens for VQAs, considered the most promising near-term quantum applications, as they run each program for thousands of iterations, where each iteration is executed for thousands of trials, and the whole iterative
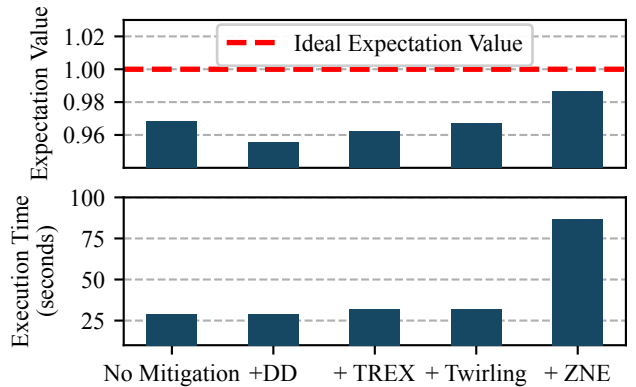

Fig. 3. Trade-offs between fidelity (expectation values) and execution latency for a 50-qubit two-local ansatz on the 127-qubit ibm_kyoto device, using five error mitigation modes: no mitigation, dynamic decoupling (DD), twirled readout error extinction (TREX), gate twirling, and zero noise extrapolation (ZNE). Error mitigation improves fidelity at the cost of increased latency.

process is repeated a few thousand times by starting with new initial program parameters (restarts).

TABLE II
AMAZON BRAKET PRICING

Provider	Device	Execution Time/Gate	Price/Task	Price/Shot
IonQ	Harmony	200 microseconds	$0.3	$0.01
IonQ	Aria	600 microseconds	$0.3	$0.03
IonQ	Forte	970 microseconds	$0.3	$0.03
Rigetti	Aspen-M	169 nanoseconds	$0.3	$0.00035

The trade-off in performance metrics is visible in the competitive cost dynamics for accessing these resources. Table II shows the cost of accessing various quantum systems on AWS Braket. Users incur an access cost to initiate a task and a shot cost based on the number of shots and per-shot price. Typically, applications run for thousands of shots. On AWS Braket, the higher-fidelity but slower Aria system costs $3\times$ more per shot than the lower-fidelity and faster Harmony [50]. More noticeably, Rigetti offers more competitive pricing, with per-shot rates $28.6\times$ to $85.7\times$ lower than IonQ, making them attractive alternatives to users, if they can be used efficiently.

2) Different systems from the same provider (same qubit technology): Systems from the same vendor using identical device technology can exhibit significant variability in error rates and gate durations. For instance, IonQ-Forte, the latest device from IonQ, offers higher fidelity but slower gate times compared to its predecessors, Aria and Harmony [50]. Similarly, ibmq_kolkata demonstrates lower noise levels than ibmq_toronto, which features an older architectural design [5]. Even within the same fabrication technology, process variation leads to differences in error characteristics. For example, two Eagle processors, ibm_kyiv and ibm_cleveland, have 1.5% and 6.6% errors per layered gate, respectively [5].

3) Single system with different software capabilities: Execution results can vary significantly even on the same system depending on the software capabilities employed. For example, the quantum volume (a benchmarking measure for quantum

systems) of ibmq_montreal increased from 64 [51] to 128 [52] within 6 months by only using software error mitigation. Error mitigation techniques, such as Dynamic Decoupling (DD) [53], Twirled readout error extinction (TREX) [54], [55], Gate Twirling [56], and Zero Noise Extrapolation (ZNE) [57], can substantially improve the fidelity of programs but often come at the cost of increased execution time and computational overheads. For example, as shown in Figure 3, applying ZNE to a 50-qubit two-local ansatz on the ibm_kyoto device reduces the error by 57-70% but also increases the execution time by $3\times$. This shows that program execution must navigate the fidelity versus execution time trade-off efficiently to achieve both high accuracy and low execution time.

The fundamental trade-off between fidelity and total execution latency worsens at the cloud level as multiple users attempt to navigate them at the application level, resulting in prolonged wait times. Qoncord optimizes for both fidelity and execution time through enhanced job scheduling. Qoncord improves the performance of VQAs when systems of multiple qubit technologies (Section III-B1), as well as systems of the same qubit technology, are used (Section III-B2).

IV. QONCORD

This paper introduces *Qoncord*, a fully automated job scheduling software for NISQ applications on quantum clouds.

A. Design Philosophy

To mitigate queuing latency, loads from a high-load device can be distributed to a low-load device. However, fully migrating a Variational Quantum Algorithm (VQA) task from a high-load device to a low-load device can introduce fidelity concerns. Typically, low-load devices exhibit lower queuing latencies due to their lower fidelity.

In response, we designed Qoncord based on the observation that VQA tasks can be divided into two distinct phases, each with varying resilience to noise. The noise-resilient phase is efficiently queued onto low-fidelity devices, and only the noise-sensitive phase is executed on the high-fidelity device. This effectively reduces the overall queuing time for the VQA task and maintains the same solution quality as if the task were executed entirely on the high-fidelity device.

B. Insight 1: Not All VQA Iterations are Equal

The VQA training process navigates the optimization landscape to converge on optimal parameter values. For example, in Figure 4, the training path is depicted on the landscape of a 7-qubit VQA benchmark using a noisy simulator. This path aims to achieve optimal parameter values for β and γ. The error model of two 27-qubit IBMQ machines, namely ibmq_toronto and ibmq_kolkata, is used for this study. We start the training process with the same initial parameter values for both machines to highlight our insight.

The parameters β and γ are tunable and are modified with each iteration using gradient descent-based classical optimizers. The gradient descent operation is employed on an optimization landscape with several symmetric optimal regions; Figure 4 shows two such symmetric optimal regions, namely one at the upper right and one at the lower left.

Bimodel Phases: The early training phase, known as the *exploration* phase, distinguishes between regions with and without optima. In this phase, the trajectory moves from the sub-optimal region in the bottom right to the optimal upper right region. Once the region with the optima is identified, a more precise parameter *fine-tuning* (shown by the dotted line), called the fine-tuning phase, locates the optima.

Observation: Relying on gradient values within the optimization landscape can enable us to identify the end of the exploration phase and the beginning of the fine-tuning phase. For instance, in Figure 4, we observe that:

1) There exist gradients on the optimization landscape for both machines. The gradients tend to saturate while the VQA task executes on the lower-fidelity ibmq_toronto device. This can point to the end of the exploration phase.

2) In the exploration phase, the optimization tends to proceed in the same direction on lower-fidelity ibmq_toronto and higher-fidelity ibmq_kolkata. Thus, this phase can be executed accurately, even on a low-fidelity device.

3) As the exploration phase concludes, the gradients tend to be sharper on the higher-fidelity ibmq_kolkata machine. This sharper gradient landscape is sufficient to distinguish between the regions with and without the optima – essentially identifying and triggering the fine-tuning phase.

4) On the contrary, on the ibmq_toronto device, the gradients are insufficient for fine-tuning. As a result, the optimization process does not converge on this device. In contrast, the fine-tuning phase is successful on ibmq_kolkata.

Approach: Starting on an inexpensive, low-latency, and lower-fidelity quantum device allows us to explore the search space quickly. We can then switch to expensive, high-latency, high-fidelity devices for fine-tuning. This can maximize resource utilization and enable faster time-to-solution.

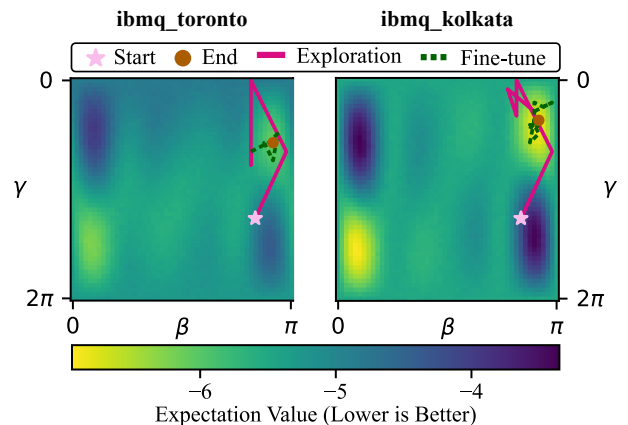### C. Insight 2: Not All Restarts Are Equal

Fig. 4. Landscape of a 7-qubit VQA and the optimizer path traced during training on two 27-qubit IBMQ systems.
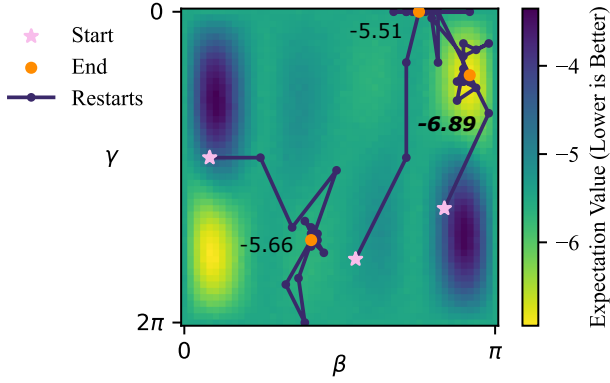
Fig. 5. Optimization paths for three restarts starting from unique initial points on the landscape of a 7-qubit VQA. Only one of the restarts converges to the global optimum (corresponding to the expectation value -6.89).

When training a VQA task, multiple rounds of optimization are required. This is because the optimization landscape can have many local optimal points in addition to the global optimum. If the training process only goes through a single round, it risks converging at one of the local optimal points instead of finding the best global solution.

Mitigating Useless Restarts: To overcome this issue, multiple training rounds or restarts are conducted, each with a distinct set of initial parameters. This approach allows each restart to explore a different region of the optimization landscape. Figure 5 illustrates this method for three separate restarts from different initial points on the landscape of a 7-qubit VQA. Even in this relatively low-dimensional space (with only two parameters), only *one* restart converges to the global optima. Typically, large circuits require as many as a hundred restarts, and our experiments observed that only 40% achieved convergence to a global optimum. Furthermore, the number of restarts increases as the number of parameters increases, posing significant computational overhead [4], [11], [28], [58].
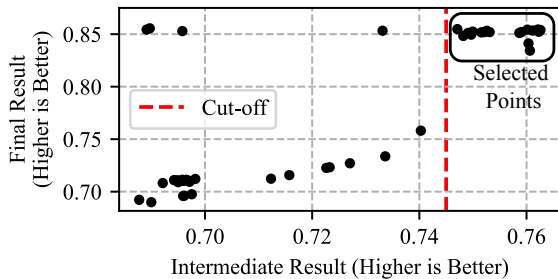

Fig. 6. "Scatter plot comparing the final result (after executing all iterations) with the intermediate result (after completing 40% of the iterations). Typically, an optimization that converges to a global optimal value also demonstrates good performance in the early stages of optimization.

Intermediate Value Clusters: One can identify high-quality restarts by observing their intermediate values. Figure 6 shows that, across various VQA circuits, the intermediate expectation values of high-quality restarts tend to be clustered. These clusters are formed by executing only 40% of the iterations per restart and lie in the exploration phase.

Approach: Qoncord reduces the restart overhead by quickly evaluating the quality of each restart using the low-fidelity, low-load device. This enables Qoncord to *only* fine-tune the high-quality restarts on high-fidelity devices.

D. Implementation Overview

Until now, we've emphasized dividing the VQA optimization into exploration and fine-tuning phases. However, a more crucial realization is that each quantum device has inherent noise, limiting its VQA optimization capabilities. Devices with lower noise and higher fidelity enhance the optimization process, leading to better solutions. Therefore, Qoncord surpasses static iteration distribution across devices.

The implementation overview of Qoncord is shown in Figure 7. Two key components facilitate dynamic scheduling in Qoncord: 1) An execution fidelity estimator that predicts the impact of device noise and ranks expected performances across devices. 2) An adaptive convergence checker setting termination conditions per device based on metrics such as expectation value and Shannon entropy.

As shown in Figure 7, Qoncord starts with the fidelity estimator to select appropriate devices for the given VQA task, then initiates exploration on the lowest-fidelity device for all restarts. The adaptive convergence checker monitors each restart, terminating low-quality restarts with intermediate values not close to a clustered set. Only high-quality restarts progress to the next higher-fidelity device for continued exploration and fine-tuning. This iterative process advances the device hierarchy toward maximum fidelity.

E. Execution Fidelity Estimator

The initial step in Qoncord is to evaluate the fidelities of available quantum devices for various stages of the optimization process. By default, Qoncord employs a metric, denoted as P_{Correct}, measuring the probability of obtaining a correct outcome on a given device. This metric, derived from prior work [26], is determined using Equation (1):

$$P_{\text{Correct}} = e^{-\frac{CD\frac{\mu_{tG_1}+\mu_{tG_2}}{2}}{T_1 T_2}}(1-\gamma)^{G_1}(1-\beta)^{G_2}(1-\omega)^M \quad (1)$$

where CD represents circuit depth, μ_{tG_1} and μ_{tG_2} denote average latencies for single and two-qubit gates, respectively. The fidelity of single-qubit gates, two-qubit gates, and measurements is represented by γ, β, and ω, respectively. $G_{1/2}$ and M indicate the number of single and two-qubit gates and the measurements count, respectively.

Minimum Estimated Fidelity: Given a VQA task and a set of available devices, Qoncord initially filters out devices that are too noisy to run the task effectively. This process involves determining a minimum acceptable circuit fidelity, as calculated in Equation 1. In Figure 8, a sample 7-qubit QAOA circuit is simulated using noise models from six IBM quantum devices. The optimization gain reported is the increase in the approximation ratio of QAOA achieved through classical optimization. While, in theory, more QAOA layers enable finding better solutions [4], i.e. with a higher optimization
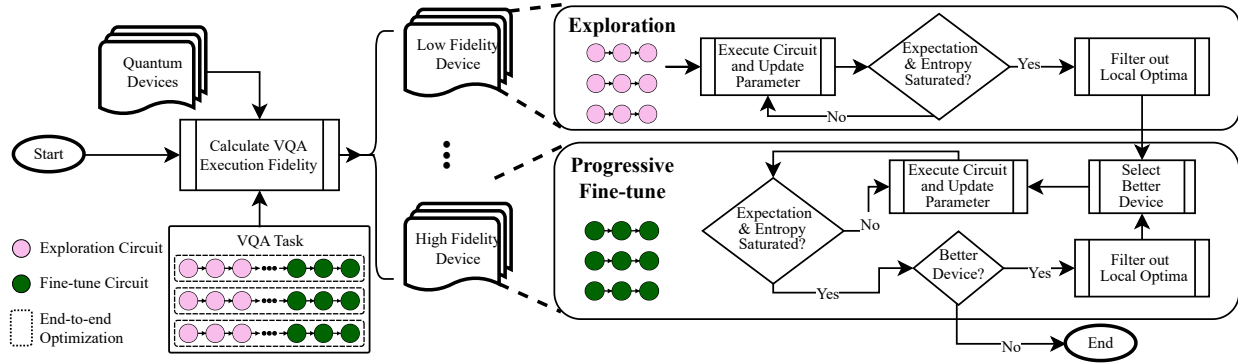
Fig. 7. Overview of Qoncord: It computes the expected execution fidelity of a given VQA task on all available devices. Then, it starts the *exploration* stage on the low-fidelity device for all restarts. Once optimization terminates, Qoncord migrates to a high-fidelity device for *fine-tuning*.
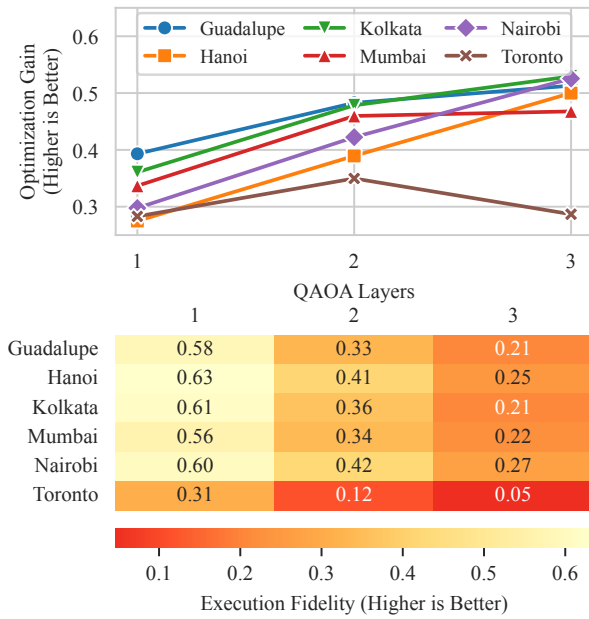

Fig. 8. Optimization performance of a 7-qubit QAOA using noise models from six IBMQ devices with up to three layers. While more layers enable better solutions (in theory), they are more error-prone on real systems due to additional gates. The heatmap shows the estimated circuit fidelity calculated using Equation 1. Fidelity below 0.1 gives poor results.

gain, additional layers also introduce more gates, leading to increased execution errors. The heatmap illustrates the estimated fidelity per device and layer count. It becomes evident that below an estimated fidelity of 0.1, adding more QAOA layers ceases to improve performance due to excessive noise, indicating a plateau in optimization progress. To address this, Qoncord establishes a minimum fidelity threshold of **0.1**. Device-task combinations with estimated fidelities below this threshold are excluded for that VQA, ensuring backend hardware can meaningfully optimize for the given task.

F. Adaptive Convergence Checker

$P_{Correct}$ offers a high-level understanding of the overall execution error rate for a specific VQA task on a given quantum device. However, this alone proves insufficient for effective VQA task scheduling across multiple devices. While $P_{Correct}$ provides valuable insights into the estimated execution fidelity of the quantum circuit running on a given device, it fails to fully capture the progress of the optimization process. This is primarily due to the fact that these metrics do not consider that parameters in the ansatz circuit of VQAs can cause variance in the final execution fidelity.

Why $P_{Correct}$ may not indicate progress: Consider the 7-qubit QAOA circuit from Figure 8 as an example. When executing on the ibmq_kolkata device with 1 QAOA layer, $P_{Correct} = 0.61$ is calculated. However, as shown in Figure 9, running the same circuit with 100 random parameter sets results in varying Hellinger fidelities [59] (from 0.56 to 0.99). The calculated $P_{Correct}$ does not align with the mean Hellinger fidelity of 0.83 which was recorded. This indicates $P_{Correct}$ cannot provide enough context on the current optimization progress as it's insensitive to parameter changes.
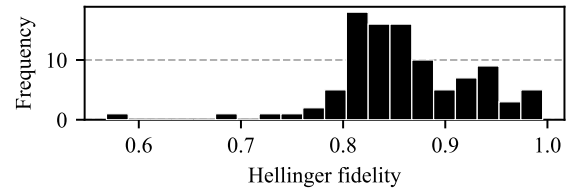

Fig. 9. Distribution of the Hellinger fidelity of a 7-qubit, 1-layer QAOA circuit running with 100 random sets of parameters. Different parameter values have different levels of noise tolerance, thus, resulting in different fidelity values.

Shannon's Entropy versus Quality of Results: The Shannon entropy [60] of the output distribution serves as another valuable metric to assess optimization progress. Entropy provides insights into the optimization trajectory beyond the expectation value. A high entropy value signals greater output randomness and more average-case results. On the other hand, a low entropy value indicates less uncertainty and is typically associated with either optimized or worst-case results.

Assess Progress using Joint Entropy and Expectation: As the VQA optimization converges, one would expect entropy to evolve from initially low (potentially a worst-case starting
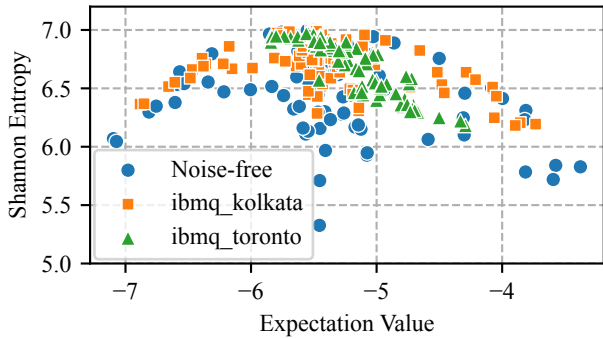
Fig. 10. Relationship between Shannon entropy and expectation values for a 7-qubit QAOA benchmark. Entropy generally anti-correlates with expectation value, but the relationship is complex. The noisy ibmq_toronto progresses from low to higher entropy states but does not converge. In contrast, the higher-fidelity ibmq_kolkata better resolves the characteristic entropy arc.

point) and transition through higher entropy average-case distributions before decreasing again as superior solutions are approached. Figure 10 visually presents this entropy arc, partially captured on noisy devices like ibmq_toronto. In contrast, higher-fidelity ibmq_kolkata demonstrates progression towards an apparent better solution. As shown in Figure 10, the same entropy value can correlate to different expectation values, and vice versa. This non-linear relationship can be exploited to avoid premature optimization termination as a single measure might indicate that no further optimization is necessary while the other still shows potential for improvement. Therefore, Qoncord utilizes both entropy and expectation values to ensure that the optimization only terminates when both metrics have stabilized at their optimal levels. When transitioning to higher-fidelity systems, Qoncord checks if entropy decreases, indicating less noisy execution, to determine whether to stay on the current device or move to the next tier.

G. Optimization Strategy in Qoncord

In Qoncord, optimizations unfold across a series of devices instead of completing end-to-end on a single machine. Before reaching the final device for fine-tuning, Qoncord employs a relaxed convergence checker with less strict criteria than the final convergence decision. For instance, if the original checker terminates optimization after *ten* iterations without improvement, the relaxed checker may instead trigger it at *five* iterations. This accounts for further exploration that may enhance the solution in subsequent stages. Only on the final device is the stringent original convergence checker applied to decide when to terminate the VQA ultimately. This two-tiered strategy prevents premature convergence while reducing unnecessary iterations on noisier devices.

H. Efficient Restart Selection

VQAs typically require multiple random restarts to avoid local optima [31]. To this end, Qoncord selectively chooses restarts to proceed to higher-quality hardware for fine-tuning. Leveraging the insight presented in Section IV-C, Qoncord

assesses the quality of restarts by examining their intermediate expectation values. Only restarts found within the top-performing cluster, as determined by this analysis, are promoted to downstream devices. Conversely, restarts deemed of low quality are terminated, enhancing overall efficiency.

I. Maintaining Up-To-Date Calibration

Quantum device calibration data in Equation 1 can become outdated. Thorough calibrations on devices are expensive and infrequent [23], [26], [61], [62]. Cloud service providers could address this transparently by periodically storing a sample of optimization outcomes and comparing new outcomes to these benchmarks. In this way, drifts in device noise profiles can be detected without added executions.

V. METHODOLOGY

A. Cloud Scheduling Policy

We compare Qoncord against the following schedulers:

1) **Least Busy:** [5], [23] This policy always selects the least occupied device for job execution, theoretically offering the highest throughput but resulting in a degraded accuracy.

2) **Load Weighted:** Devices are chosen based on their load, with less loaded machines being more likely to be selected.

3) **Fidelity Weighted:** Devices are selected based on their fidelity, which is the general user access patterns.

4) **Best Fidelity:** [23] This policy always selects one of the highest fidelity devices, aiming for the best possible results but potentially incurring longer latencies due to the limited number of high-performance devices.

5) **EQC:** [26] This policy involves the ensemble execution of variational quantum algorithms across all devices. It increases the total number of circuits to be executed. In our evaluation, the EQC scheduling is modeled by converting runtime jobs into individual tasks and using the least-busy method for scheduling. Each runtime job under this model requires twice the number of tasks compared to other methods, representing the minimum overhead for a 1-layer QAOA.

> Ideally, the wait time of the *Least Busy* machine and fidelity of the *Best Fidelity* machine is desired.

B. Figure of Merit

The evaluation focuses on system throughput and accuracy.
1. Throughput: Throughput is the number of tasks completed per unit of time. To evaluate throughput in our study, we calculate it using the formula:

$$\text{Throughput} = \frac{\text{Number of Circuits}}{\text{Completion Time}} \quad (2)$$

This metric allows us to compare the efficiency of different scheduling methods under identical conditions. It is important to note that for the EQC policy, the number of circuits is higher relative to other policies. A detailed discussion on EQC scheduling is provided in Section VI-G.

2. Accuracy: To assess accuracy, we evaluate the performance of VQAs using *approximation ratio*. It compares the optimized expectation value ($E_{\text{optimized}}$) derived from the VQA to the ground truth expectation value ($E_{\text{ground\_truth}}$) obtained through brute force searching and is the theoretical minimum expectation value. Mathematically, this ratio is expressed as:

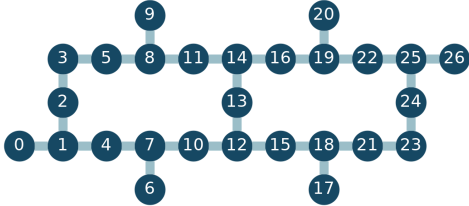$$\text{Approximation Ratio} = \frac{E_{\text{optimized}}}{E_{\text{ground\_truth}}} \qquad (3)$$

Fig. 11. Coupling map of the 27-qubit IBM devices.

C. VQA Algorithms

We assess Qoncord using two prominent VQAs: Quantum Approximated Optimization Algorithm (QAOA) [4] and Variational Quantum Eigensolver (VQE) [28]. For QAOA, we focus on the max-cut problem with two Erdős Rényi random graphs [63]: one with seven and the other with nine nodes. Both graphs are generated with an edge probability of 0.5. We examine QAOA circuits with 1, 2, and 3 layers. For VQE, we choose the hydrogen molecule (H_2) and use a 4-qubit unitary coupled cluster single and double excitation (UCCSD) [64] ansatz to find the ground state energy of H_2. All the circuits are transpiled using qiskit transpiler with O3 optimization.

D. Noisy Simulation Setup

By default, we use device error profiles from ibmq_kolkata and ibmq_toronto. As shown in Figure 11, both devices have the same coupling map. ibmq_kolkata (high-fidelity device) has an average 2-qubit gate error rate of 1.091% and readout error rate of 1.22%. ibmq_toronto (low-fidelity device), on the other hand, has a higher average 2-qubit gate error rate of 2.083% and readout error rate of 4.48%. The 36-qubit IonQ-Forte device noise profile is also used for sensitivity studies. It has an all-to-all connection with an average 2-qubit gate error of 0.74% and an average readout error of 0.5%.

E. Execution Platform

We use Qiskit [65] 0.45.0 for the circuit simulation and optimization trajectory analysis. We use noise models generated from ibmq_toronto, ibmq_kolkata, and IonQ- Forte for noisy simulations. Circuit simulations are run on a cluster with a 40-core Intel Xeon Gold 6230 processor nodes (2.1GHz with 192GB DDR4-2666 ECC memory). We use Qiskit's implementation of the Simultaneous Perturbation Stochastic Approximation (SPSA) [66] for the classical optimizer.

F. Scheduling Simulation Setup

We compare scheduling policies discussed in Section V-A. Our experiment aims to replicate real-world conditions and involves a pseudo workload of 1000 quantum jobs. These jobs include independent tasks, executed once, and runtime jobs, which account for 10 to 90% of all jobs, that continuously submit circuit executions during an active session. Variable delays separate the consecutive executions within a runtime session to mimic the behavior of runtime job scheduling in real-world workloads [24]. This allowed for the insertion of other queued jobs. Execution times randomly vary 3× between minimum and maximum, reflecting empirical hardware behavior [23]. Tests cover hardware with an execution fidelity between 0.3–0.9.

VI. RESULTS AND ANALYSIS
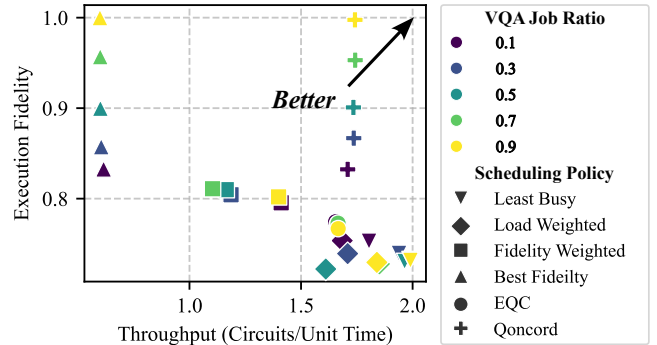
A. Performance Analysis of Scheduling Policies

Fig. 12. Fidelity-throuput analysis. We evaluate different scheduling policies against Qoncord using a simulation of 1000 quantum jobs on 10 hypothetical devices with fidelities ranging from 0.3 to 0.9. Qoncord is the closest to an ideal policy as all its points lie closer to the top right-hand corner.

Figure 12 shows the fidelity relative to the highest-fidelity device and execution throughput. Qoncord consistently delivers high fidelity while having a high throughput that is very close to the `Least Busy` policy even when the device fidelity gap alters due to temporal variations in error-rates. framework outperforms other policies as the percentage of VQA jobs or total cloud load increases. This capability is essential for VQAs, which demand both accuracy and efficiency. It also enables cloud providers to reduce the time to access machines (and time to solution) without compromising accuracy. Policies such as `Least Busy` and `Load Balanced` achieve higher fidelities but also significantly compromise fidelity. The `Fidelity Weighted` policy does not excel in either metric. EQC scheduling, despite operating on a least busy principle, faces challenges due to the 2× circuit execution (discussed more in Section VI-G) overhead of VQA tasks. This overhead resulted in only moderate improvements in throughput. Note that a direct assessment of the average fidelity of EQC is not applicable because, unlike other scheduling strategies, including Qoncord, EQC employs an *asynchronous* gradient descent strategy. This aspect of EQC is discussed more in detail in Section VI-G.

9

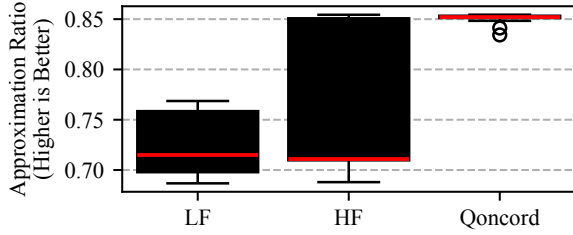## B. End-to-End Multi-Restart VQA Optimization

Fig. 13. Distribution of approximation ratios across 50 random QAOA optimization restarts. Qoncord starts optimization on the LF device and fine-tunes on the HF device. Overall, Qoncord matches the maximum approximation ratio achieved of HF-only optimization and its result, on average, is at least 20% higher than any of the single-device optimization results.

We evaluate the impact of Qoncord on restarts required for VQAs. Figure 13 shows the approximation ratios achieved by Qoncord, the HF device alone, and the LF device alone across 50 random restarts for a 3-layer QAOA benchmark. Qoncord filters out 31 bad-performing optimization runs when transitioning from LF to HF device and only the remaining 19 subsequently progressed on the HF device. As a result, Qoncord produces a mean approximation ratio over 20% higher than the other cases.
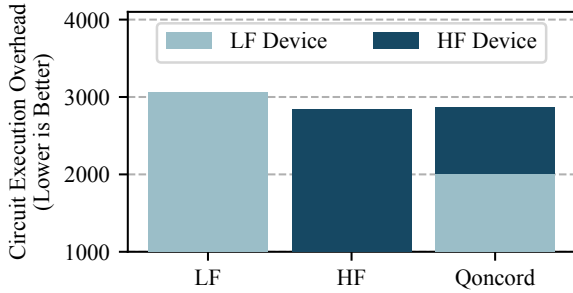

Fig. 14. Circuit execution overhead for 50 random QAOA restarts across three execution modes: LF-only, HF-only, and Qoncord. All three modes incur similar overall circuit execution overheads. However, Qoncord distributes its executions evenly across available devices, this greatly optimizes the overall resource usage and enhances execution efficiency.

While providing a result with higher quality, Qoncord also reduces the load on each device. Figure 14 shows the circuit execution overheads of using LF-device only, HF-device only, and Qoncord. In this example, using only the LF device requires 3,055 circuit executions, whereas using the HF device alone requires 2,841 circuit executions. Qoncord, on the other hand, requires a total of 2,865 circuit executions, out of which 2,009 are executed on the LF device and the remaining 856 on the HF device. This means the LF-device executes 70% of the total executions while also ensuring greater accuracy due to early termination of poor optimization runs.

C. Multi-Restart VQA Optimization: More Quantum Devices

Qoncord is designed to integrate additional devices into its execution seamlessly and can iteratively enhance the outcomes
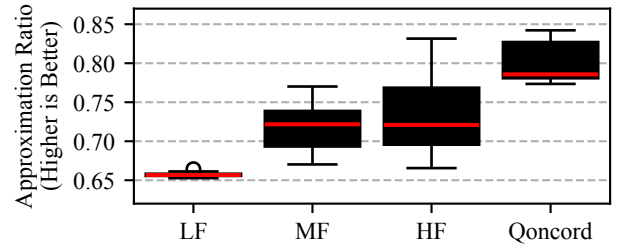

Fig. 15. Distribution of approximation ratios across 50 random QAOA optimization restarts. Three device noise profiles are included, ibmq_toronto is used as low fidelity (LF), ibmq_kolkata medium fidelity (MF), and IonQ-Forte high fidelity (HF). Qoncord starts optimization on the LF device and progressively moves to higher fidelity devices to enhance the execution result.
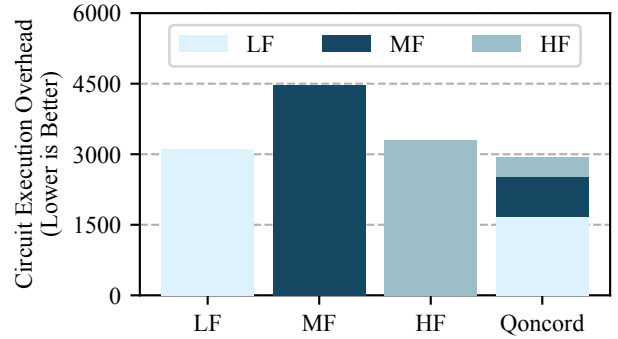

Fig. 16. Circuit execution overhead for 50 random QAOA restarts across four execution modes: LF-only, MF-only, HF-only, and Qoncord. LF and HF modes, along with Qoncord, show similar, lower overheads as optimizers quickly converge in very noisy or nearly noise-free environments. In contrast, the MF-only mode exhibits higher overheads, this suggests that with moderate noise levels, significant improvements are possible but harder to achieve.

of multi-restart optimizations performed with them. We have incorporated three device profiles for demonstration and used them to optimize a 9-qubit, 3-layer QAOA circuit. Specifically, ibmq_toronto is used as low fidelity (LF), ibmq_kolkata medium fidelity (MF), and IonQ-Forte high fidelity (HF). Figure 15 shows the distribution of 50 random restarts of the VQA task performed individually on each device and with Qoncord. Notably, Qoncord provides the highest approximation ratio achieved. Furthermore, the average performance of Qoncord across all restarts also significantly exceeds that of any single-device setup, with a more than 8% improvement in average approximation ratio over all single-device executions.

Figure 16 shows the corresponding circuit execution overheads. Overall, LF, HF, and Qoncord modes exhibit similar, lower execution overheads. This indicates that Qoncord effectively leverages the available devices by balancing the computational load, thus reducing the demand for higher-fidelity devices. In contrast, the MF-only mode shows significantly higher execution overheads, suggesting that moderate noise levels present a more challenging optimization landscape.

In essence, Qoncord not only matches the performance of high-fidelity devices but does so with enhanced efficiency by distributing executions in a way that leverages the strengths of each device. This strategic management allows it to maintain

high accuracy without placing undue strain on any single device, particularly those with higher fidelity.

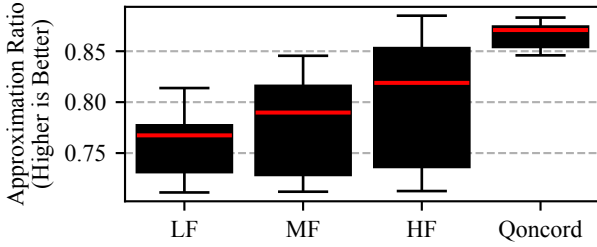## D. Multi-Restart VQA Optimization: Larger Quantum Circuits

Fig. 17. Distribution of approximation ratios across 50 random QAOA optimization restarts for a 14-qubit 1-layer QAOA task.
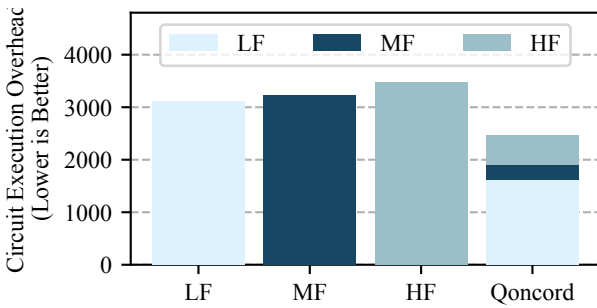

Fig. 18. Circuit execution overhead for the 14-qubit QAOA optimization task.

We tested Qoncord on a 14-qubit VQA job, the largest simulatable using density matrix-based noisy quantum circuit simulators on modern GPUs. As no existing quantum device noise profiles met our minimum fidelity criteria (Section IV-E), we created three hypothetical noise models with depolarization error rates for 2-qubit gates and readout: 0.1% (high-fidelity), 0.5% (mid-fidelity), and 1% (low-fidelity). Figure 17 shows approximation ratios for these models and Qoncord. Even at this increased complexity, Qoncord outperformed single-device results. Figure 18 displays corresponding circuit execution overheads. Qoncord effectively utilized low and mid-fidelity devices while showing better results.

E. Single Restart QAOA Optimization

We also analyzed the performance of Qoncord on a single VQA optimization restart *without* early termination. We used a 3-layer QAOA circuit and compared it against running the full optimization simulated using noise profiles from the ibmq_kolkata (HF) and ibmq_toronto (LF) devices. In theory, running solely on the HF device should produce the best approximation ratio. Our goal with Qoncord is to achieve a comparable result as the HF device while reducing the load placed on that device. Figure 19 shows that Qoncord achieves an approximation ratio very close to the HF-only case and over 14% higher than the LF-only optimization.

Furthermore, by dynamically switching between the HF and LF executions, Qoncord reduces the load on each individual
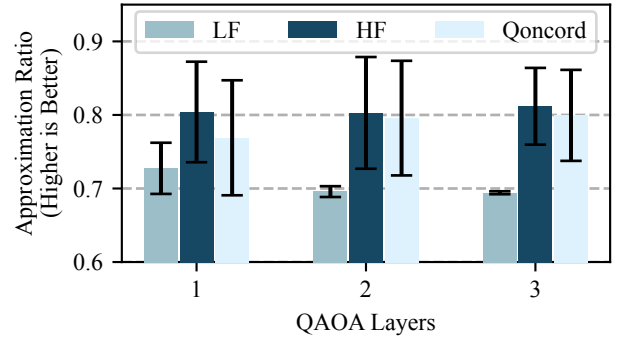

Fig. 19. Approximation ratio for a single QAOA optimization restart. Qoncord achieves similar performance as the HF-only case.
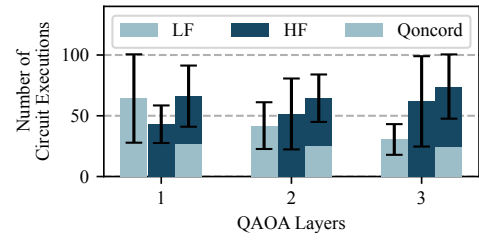

Fig. 20. Average number of circuit executions for a single QAOA restart comparison. While the total executions are similar, Qoncord reduces the executions on the individual HF and LF devices.

device. As seen in Figure 20, Qoncord required similar total executions as the HF and LF simulations but with fewer executions on either simulated device alone. By balancing computations across devices, Qoncord can achieve results comparable to an HF-only optimization with a lower peak load. This shows the capability to maintain accuracy while mitigating bottlenecks when scaling to larger optimizations.
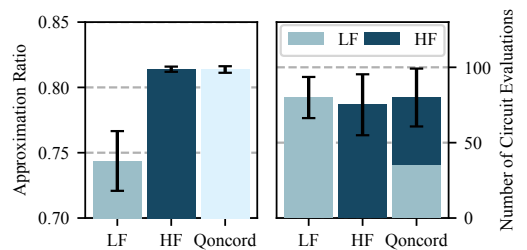
F. VQA Optimization of VQE

Fig. 21. Accuracy and execution overheads for a 4-qubit VQE running with hydrogen molecule and UCCSD ansatz. Qoncord matches HF accuracy with no additional executions beyond those needed for HF or LF optimizations.

We evaluate Qoncord on a VQE application using a 4-qubit UCCSD ansatz to find the ground state energy of a hydrogen molecule. As with the QAOA testing, noise profiles from HF (ibmq_kolkata) and LF (ibmq_toronto) devices were used. As shown in Figure 21, Qoncord achieves a ground state energy within 0.3% of the HF-only optimization. By dynamically allocating executions across the HF and LF devices, Qoncord

introduces almost no additional executions beyond those required for HF or LF alone. Qoncord matches the accuracy of an HF optimization while reducing the computational load on that device. This highlights the benefits of using Qoncord.

G. Case Study: Asynchronous Gradient Descent

We compare our Qoncord against the asynchronous gradient descent (AGD) optimization used in EQC [26]. EQC optimizes individual parameters on different devices separately, and the results are combined at each epoch's end. Qoncord differs by optimizing all parameters together across multiple devices. We evaluate both approaches using a 3-layer QAOA circuit. Figure 22 shows that even a single AGD epoch requires more executions than optimizing all parameters together on the HF device. Also, the approximation ratio after 1 AGD epoch is much lower than Qoncord or HF only. EQC results also show slower convergence when distributing optimizations across multiple devices compared to using only a high-fidelity device due to the limitations of the low-fidelity devices. In contrast, Qoncord is not constrained by averaging intermediate results and fully leverages both devices throughout the execution. Overall, the synchronous optimization approach in Qoncord is better suited for leveraging multiple devices than AGD.
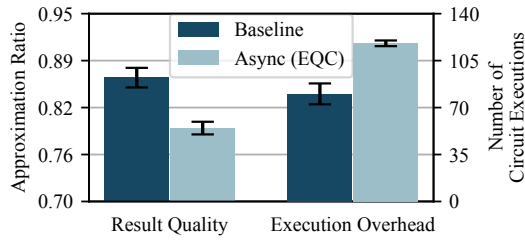

Fig. 22. Accuracy and executions after one epoch of asynchronous gradient descent (AGD) on a 3-layer QAOA circuit. AGD requires more executions than optimizing all parameters together and achieves lower accuracy.

VII. Related Works

A. Improving Utilization In Quantum Clouds

Improving the utilization of devices in quantum clouds has been previously studied. Recent works have emphasized multiprogramming to enhance throughput and resource utilization. For instance, Das et al. [67] propose concurrent execution of multiple quantum workloads on NISQ machines. Building on this concept, Resch et al. [68] extend the idea to study circuit-level concurrency specifically for VQA circuits. On the other hand, QuCloud [69] focuses on optimizing qubit mapping to efficiently allocate quantum resources among various programs in a cloud environment. Similarly, CutQC [70] introduces a hybrid classical-quantum computing method, partitioning large quantum circuits to execute on smaller quantum devices. These approaches, focusing on throughput enhancement, are orthogonal to Qoncord, which explores leveraging different quantum devices at various optimization stages of VQAs. Combining these methodologies could lead to more effective and efficient quantum cloud computing paradigms.

B. VQA Optimization

Training VQAs poses substantial execution overheads. Prior work highlights a clustering phenomenon in QAOA [71], indicating that optimal parameters from one problem can be transferred to others, providing near-optimal outcomes [72], [73]. Conversely, CAFQA [74] employs classical simulation to optimize VQE, using a Clifford circuit variant that is classically simulatable. However, this type of warm start initialization does not apply to QAOA [75]. Ansatz optimization is also crucial for enhancing algorithmic efficiency. AdaptVQE [76], for instance, dynamically constructs the ansatz during the execution to minimize the depth of the ansatz circuit.

C. Hybrid Optimization

In classical optimization, an extensive set of studies focused on hybrid models that strategically combine different techniques to improve performance. Prior work [77] proposes a technique merging fast and frugal decision trees with machine learning models. This hybrid approach aims to couple the interpretability of trees with the superior accuracy of neural networks. Similarly, Kudva et al. [78] investigate constrained Bayesian optimization algorithms for tuning noisy blackbox functions. By incorporating robust stochastic models within an efficient sequential design strategy, their method provides regret bounds unmatched by either paradigm alone. Beyond marrying simpler and more sophisticated techniques, researchers also envision hybrids of different cutting-edge approaches. He et al. [79] review recent work on surrogate-assisted evolutionary algorithms capable of optimizing extremely expensive objective functions. Gaussian process-based Bayesian optimization has proven highly effective at handling single and multi-objective design problems involving complex systems. If these classical optimization methods are suitable for the exploration phase of VQA optimization, they could be integrated with Qoncord, as most optimization approaches comprise initial exploration followed by fine-tuning.

VIII. Conclusion

Quantum cloud services are essential for enabling remote access to quantum devices, but they struggle with a significant demand-supply mismatch and uneven load distribution due to varying device fidelity. As a result, program execution is either impacted by the wait times of the high-fidelity devices or the noise of the low-fidelity devices. This paper proposes *Qoncord*, a fully automated job scheduling framework that splits the program execution into phases with varying noise resilience. Qoncord schedules the more noise-resilient phase on the low-fidelity device (for higher execution throughput). It runs the more noise-sensitive phase on the high-fidelity device (for higher application fidelity). Qoncord also leverages the insight that near-term quantum applications comprise several independent training rounds that start from different initial points, but not all of them lead to the optimal solution. Qoncord quickly evaluates these candidates on the low-latency device, eliminates the weaker candidates, and runs the remaining ones on the high-fidelity machine. Overall, Qoncord offers similar

solutions 17.4 × faster than the baseline, or it provides 13.3% better solutions on average with a similar execution time.

Appendix A: Artifact Appendix

A. Abstract

Our artifacts include the Qoncord job scheduling framework for Variational Quantum Algorithms (VQAs), designed for cloud-based Noisy Intermediate-Scale Quantum (NISQ) systems but primarily tested using classical noisy quantum circuit simulators. Qoncord optimizes the execution of VQAs by strategically dividing the training process into exploratory and fine-tuning phases, and distributing these across different quantum devices based on their fidelity and availability. The framework includes algorithms for adaptive scheduling and optimization of restart procedures. Our artifacts demonstrate Qoncord's ability to achieve solutions 17.4× faster than baseline approaches and deliver 13.3% better solutions within the same time budget using simulated NISQ environments.

B. Artifact check-list (meta-information)

- **Hardware:** Classical simulation can be running on both CPU and GPU platforms
- **Metrics:** Execution overhead, load balancing, solution quality
- **Output:** Optimized VQA (QAOA and VQE) solutions from simulated quantum environments
- **Experiments:** Comparison of Qoncord vs. baseline scheduling approaches using quantum circuit simulators for QAOA and VQE problems
- **How much time is needed to prepare workflow (approximately)?:** Less than 10min
- **How much time is needed to complete experiments (approximately)?:** About 30 minutes
- **Publicly available?:** Yes
- **Code licenses (if publicly available)?:** MIT license
- **Workflow automation framework used?:** No
- **Archived (provide DOI)?:** N/A

C. Description

1) How to access: The Qoncord framework and associated code for our experiments are publicly available via GitHub at: https://github.com/meng-ubc/Qoncord

2) Hardware dependencies: Our experiments were primarily conducted using classical hardware to simulate noisy quantum circuits. While Qoncord is designed for cloud-based NISQ systems, no actual quantum hardware is required to run our simulations. There are no strict hardware requirements for running the code, but for optimal execution time, we recommend at least the following setup:

- **CPU**: 8-core processor
- **RAM**: 16 GB
- **GPU**: Recommended for noisy simulation experiments with more than 10 qubits
- **Storage**: No particular requirements; the code and data files occupy less than tens of megabytes

Note that while these specifications are recommended for better performance, the code can run on less powerful machines, which will take longer execution times.

3) Software dependencies: To run our code and reproduce the experiment results presented in the paper, the following software dependencies are required:

- **Quantum Computing**: qiskit, qiskit-ibm-runtime, qiskit-aer
- **Data Visualization**: matplotlib, seaborn
- **Additional Libraries**: numpy, scipy, networkx, tqdm

A complete list of dependencies with specific versions that have been tested can be found in the `requirements.txt` file in our repository.

4) Data sets: Our experiments use two types of VQAs: QAOA with randomly generated NetworkX graphs, and VQE with the quantum observable of a hydrogen molecule using the UCCSD ansatz. Detailed information about these can be found in the methodology section.

For the noise model, we utilized the fake backend feature from Qiskit, which incorporates noise profiles that were profiled on real quantum devices.

D. Installation

To install Qoncord and its dependencies, follow these steps:

1) Clone the repository:

```
git clone https://github.com/ \
meng-ubc/Qoncord.git
cd Qoncord
```

2) Create a virtual environment (optional but recommended):

```
conda create -n qoncord_env
conda activate qoncord_env
```

3) Install the required dependencies:

```
pip install -r requirements.txt
```

E. Experiment workflow

To facilitate the reproduction of our key results, we have prepared a Python script for each experiment. Running any of these scripts is straightforward—simply execute the command `python script.py`. This will automatically run the corresponding experiment and save the results to a file. Then a plotting script can be used to generate the relevant plots.

Each figure produced by these scripts can be directly compared against the figures presented in the paper, ensuring an easy and accurate validation of the results.

F. Evaluation and Expected Results

This section provides a detailed evaluation of each experiment. Each subsubsection introduces the experiment, specifies the Python script to be executed, identifies the corresponding results in the paper, outlines the expected runtime, and briefly explains how to interpret the output.

*1) **Queue Simulation**:* This experiment simulates a quantum queue to evaluate different scheduling policies under various VQA job ratios. The goal is to analyze the trade-offs between execution fidelity and throughput.

Script: `main_queue_sim.py`
Paper Results: Section VI.A, specifically Figure 12.
Expected Runtime: Approximately 2 minutes.

Expected Results: A scatter plot showing execution fidelity versus throughput for all scheduling policies and VQA job ratios. This plot can be directly compared to Figure 12.

*2) **Multi-Restart QAOA Optimization**:* This experiment evaluates the performance of different multi-restart strategies for QAOA optimization and compares their circuit execution overhead across various quantum devices.

Script: `2_qaoa_optimization.py`
Paper Results: Section VI.B, specifically Figures 13, 14.
Expected Runtime: Approximately 10 minutes.

Expected Results: A box plot showing the distribution of the approximation ratio for different approaches. Also, a bar plot displaying the circuit execution overhead on each device. These plots correspond to Figures 13 and 14 in the paper.

*3) **Single-Restart VQE Optimization**:* This experiment focuses on optimizing VQE using a single-restart approach, comparing the performance and execution overhead of different execution modes.

Script: `3_vqe_optimization.py`
Paper Results: Section VI.F, specifically Figure 21.
Expected Runtime: Approximately 4 minutes.

Expected Results: A bar plot showing the approximation ratio and circuit execution overhead for different execution modes, which can be compared to Figure 21 in the paper.

*4) **Comparison to Asynchronous Gradient Descent**:* This experiment compares the performance of the default gradient descent method with an asynchronous gradient descent approach in optimizing quantum circuits.

Script: `4_ae_async.py`
Paper Results: Section VI.G, specifically Figure 22.
Expected Runtime: Approximately 6 minutes.

Expected Results: A bar plot showing the approximation ratio and circuit execution overhead for both default and asynchronous gradient descent methods.

REFERENCES

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[2] S. Lloyd, "Universal quantum simulators," *Science*, vol. 273, no. 5278, pp. 1073–1078, 1996.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[4] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[5] "IBM Quantum," https://quantum-computing.ibm.com/, 2021, [Accessed: 2023-11-04].

[6] "Amazon Braket," https://aws.amazon.com/braket/, 2020, [Accessed: 2023-11-04].

[7] "Azure Quantum," https://azure.microsoft.com/en-ca/products/quantum, 2019, [Accessed: 2023-11-04].

[8] IBM, "The ibm quantum development roadmap," https://www.ibm.com/quantum/roadmap.

[9] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[10] S. S. Tannu, Z. A. Myers, P. J. Nair, D. M. Carmean, and M. K. Qureshi, "Taming the instruction bandwidth of quantum computers via hardware-managed error correction," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 679–691. [Online]. Available: https://doi.org/10.1145/3123939.3123940

[11] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, "The theory of variational hybrid quantum-classical algorithms," *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[12] F. Hua, M. Wang, G. Li, B. Peng, C. Liu, M. Zheng, S. Stein, Y. Ding, E. Z. Zhang, T. Humble, and A. Li, "Qasmtrans: A qasm quantum transpiler framework for nisq devices," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1468–1477. [Online]. Available: https://doi.org/10.1145/3624062.3624222

[13] M. Wang, B. Fang, A. Li, and P. J. Nair, "Red-qaoa: Efficient variational optimization through circuit reduction," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 980–998. [Online]. Available: https://doi.org/10.1145/3620665.3640363

[14] M. Wang, B. Fang, A. Li, and P. Nair, "Efficient qaoa optimization using directed restarts and graph lookup," in *Proceedings of the 2023 International Workshop on Quantum Classical Cooperative*, ser. QCCC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 5–8. [Online]. Available: https://doi.org/10.1145/3588983.3596680

[15] M. Wang, R. Huang, S. Tannu, and P. Nair, "Tqsim: A case for reuse-focused tree-based quantum circuit simulation," 2022. [Online]. Available: https://arxiv.org/abs/2203.13892

[16] M. Wang, F. Hua, C. Liu, N. Bauman, K. Kowalski, D. Claudino, T. Humble, P. Nair, and A. Li, "Enabling scalable vqe simulation on leading hpc systems," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1460–1467. [Online]. Available: https://doi.org/10.1145/3624062.3624221

[17] A. Li, C. Liu, S. Stein, I.-S. Suh, M. Zheng, M. Wang, Y. Shi, B. Fang, M. Roetteler, and T. Humble, "Tanq-sim: Tensorcore accelerated noisy quantum system simulation via qir on perlmutter hpc," 2024. [Online]. Available: https://arxiv.org/abs/2404.13184

[18] Alibaba Cloud, "Quantum computing: A brief overview," https://www.alibabacloud.com/knowledge/hot/quantum-computing-a-brief-overview.

[19] Xanadu, "Quantum computational advantage," https://www.xanadu.ai/.

[20] Google Quantum AI, "Google quantum computing service," https://quantumai.google/cirq/google/concepts.

[21] Terra Quantum, "Quantum as a service," https://terraquantum.swiss/quantum-as-a-service.

[22] D-Wave Leap, "The only real-time quantum cloud service built for business," https://www.dwavesys.com/solutions-and-products/cloud-platform.

[23] G. S. Ravi, K. N. Smith, P. Gokhale, and F. T. Chong, "Quantum computing in the cloud: Analyzing job and machine characteristics," in *2021 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2021, pp. 39–50.

[24] B. Johnson, "Qiskit runtime, a quantum-classical execution platform for cloud-accessible quantum computers," in *APS March Meeting Abstracts*, vol. 2022, 2022, pp. T28–002.

[25] IBM, "Qiskit runtime ibm client," https://docs.quantum-computing.ibm.com/api/qiskit-ibm-runtime/qiskit_ibm_runtime.Session.

[26] S. Stein, N. Wiebe, Y. Ding, P. Bo, K. Kowalski, N. Baker, J. Ang, and A. Li, "Eqc: ensembled quantum computing for variational quantum algorithms," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 59–71.

[27] IBM, "Quantum protein folding algorithms," https://protein-folding-demo.mybluemix.net/, year=2016.

[28] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[29] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[30] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, p. 4812, 2018.

[31] R. Shaydulin, K. Marwaha, J. Wurtz, and P. C. Lotshaw, "Qaoakit: A toolkit for reproducible study, application, and verification of the qaoa," in *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*. IEEE, 2021, pp. 64–71.

[32] IBM Quantum, "Boeing seeks new ways to engineer strong, lightweight materials," https://www.ibm.com/case-studies/boeing.

[33] IBMQ, "JSR envisions a revolution in semiconductor manufacturing," https://www.ibm.com/case-studies/jsr.

[34] IBM Quantum, "ExxonMobil strives to solve complex energy challenges," https://www.ibm.com/case-studies/exxonmobil.

[35] IBMQ, "In quantum pursuit of game-changing power sources," https://www.ibm.com/case-studies/mitsubishi-chemical.

[36] IBM Quantum, "Architecting molecules that redefine luminescence," https://www.ibm.com/case-studies/jsr-mitsubishi-keio/.

[37] IBM, "Envisioning a new wave in power: Mercedes-Benz bets on quantum to craft the future of electric vehicles," https://www.ibm.com/case-studies/daimler.

[38] IBMQ, "The quest to understand what sews the universe together," https://www.ibm.com/case-studies/cern/.

[39] Microsoft, "Azure Quantum Network," https://azure.microsoft.com/en-us/solutions/quantum-computing/network#solution-partners.

[40] Amazon, "Amazon Braket Customers," https://aws.amazon.com/braket/customers/.

[41] Accenture Labs, "Accenture Labs works with Biogen to apply quantum computing to accelerate drug discovery," https://www.accenture.com/us-en/case-studies/life-sciences/quantum-computing-advanced-drug-discovery.

[42] Xanadu Press, "Multiverse Computing partners with Xanadu to deliver quantum software solutions for finance," https://www.xanadu.ai/press/multiverse-computing-partners-with-xanadu-to-deliver-quantum-software-solutions-for-finance.

[43] Quantinuum News, "BMW Group, Airbus and Quantinuum Collaborate to Fast-Track Sustainable Mobility Research Using Cutting-Edge Quantum Computers," https://www.quantinuum.com/news/bmw-group-airbus-and-quantinuum-collaborate-to-fast-track-sustainable-mobility-research-using-cutting-edge-quantum-computers.

[44] Quantinuum, "HSBC and Quantinuum Explore Real World Use Cases of Quantum Computing in Financial Services," https://www.quantinuum.com/news/hsbc-and-quantinuum-explore-real-world-use-cases-of-quantum-computing-in-financial-services.

[45] IBM Quantum, "What is Qiskit Runtime?" 2022, https://www.youtube.com/watch?v=gSK3XRuLKB4.

[46] IonQ, "IonQ Algorithmic Qubits (#AQ)," https://ionq.com/algorithmic-qubits.

[47] J.-S. Chen, E. Nielsen, M. Ebert, V. Inlek, K. Wright, V. Chaplin, A. Maksymov, E. Páez, A. Poudel, P. Maunz *et al.*, "Benchmarking a trapped-ion quantum computer with 29 algorithmic qubits," *arXiv preprint arXiv:2308.05071*, 2023.

[48] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Physical Review A*, vol. 100, no. 3, p. 032328, 2019.

[49] Rigetti, "Rigetti QCS," https://qcs.rigetti.com/dashboard.

[50] IonQ, "IonQ Quantum Cloud," https://cloud.ionq.com/backends.

[51] P. Jurcevic, A. Javadi-Abhari, L. S. Bishop, I. Lauer, D. F. Bogorin, M. Brink, L. Capelluto, O. Günlük, T. Itoko, N. Kanazawa *et al.*, "Demonstration of quantum volume 64 on a superconducting quantum computing system," *Quantum Science and Technology*, vol. 6, no. 2, p. 025020, 2021.

[52] P. Jurcevic, D. Zajac, J. Stehlik, I. Lauer, and R. Mandelbaum, "Pushing quantum performance forward with our highest quantum volume yet," https://www.ibm.com/quantum/blog/quantum-volume-256, 2022, accessed: 2024-06-20.

[53] L. Viola and S. Lloyd, "Dynamical suppression of decoherence in two-state quantum systems," *Physical Review A*, vol. 58, no. 4, p. 2733, 1998.

[54] F. B. Maciejewski, Z. Zimborás, and M. Oszmaniec, "Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography," *Quantum*, vol. 4, p. 257, 2020.

[55] S. Bravyi, S. Sheldon, A. Kandala, D. C. Mckay, and J. M. Gambetta, "Mitigating measurement errors in multiqubit experiments," *Physical Review A*, vol. 103, no. 4, p. 042605, 2021.

[56] J. Emerson, M. Silva, O. Moussa, C. Ryan, M. Laforest, J. Baugh, D. G. Cory, and R. Laflamme, "Symmetrized characterization of noisy quantum processes," *Science*, vol. 317, no. 5846, pp. 1893–1896, 2007.

[57] K. Temme, S. Bravyi, and J. M. Gambetta, "Error mitigation for short-depth quantum circuits," *Physical review letters*, vol. 119, no. 18, p. 180509, 2017.

[58] X. Yuan, S. Endo, Q. Zhao, Y. Li, and S. C. Benjamin, "Theory of variational quantum simulation," *Quantum*, vol. 3, p. 191, 2019.

[59] E. Hellinger, "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen." *Journal für die reine und angewandte Mathematik*, vol. 1909, no. 136, pp. 210–271, 1909.

[60] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[61] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Randomized benchmarking of quantum gates," *Physical Review A*, vol. 77, no. 1, p. 012307, 2008.

[62] IBM Quantum, "About calibration jobs," https://docs.quantum-computing.ibm.com/admin/calibration-jobs.

[63] P. ERDdS and A. R&wi, "On random graphs i," *Publ. math. debrecen*, vol. 6, no. 290-297, p. 18, 1959.

[64] R. J. Bartlett and M. Musiał, "Coupled-cluster theory in quantum chemistry," *Reviews of Modern Physics*, vol. 79, no. 1, p. 291, 2007.

[65] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," 2023.

[66] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[67] P. Das, S. S. Tannu, P. J. Nair, and M. Qureshi, "A case for multi-programming quantum computers," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 291–303.

[68] S. Resch, A. Gutierrez, J. S. Huh, S. Bharadwaj, Y. Eckert, G. Loh, M. Oskin, and S. Tannu, "Accelerating variational quantum algorithms using circuit concurrency," *arXiv preprint arXiv:2109.01714*, 2021.

[69] L. Liu and X. Dou, "Qucloud: A new qubit mapping mechanism for multi-programming quantum computing in cloud environment," in *2021 IEEE International symposium on high-performance computer architecture (HPCA)*. IEEE, 2021, pp. 167–178.

[70] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, "Cutqc: using small quantum computers for large quantum circuit evaluations," in *Proceedings of the 26th ACM International conference on architectural support for programming languages and operating systems*, 2021, pp. 473–486.

[71] F. G. Brandao, M. Broughton, E. Farhi, S. Gutmann, and H. Neven, "For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances," *arXiv preprint arXiv:1812.04170*, 2018.

[72] A. Galda, X. Liu, D. Lykov, Y. Alexeev, and I. Safro, "Transferability of optimal qaoa parameters between random graphs," in *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2021, pp. 171–180.

[73] R. Shaydulin, P. C. Lotshaw, J. Larson, J. Ostrowski, and T. S. Humble, "Parameter transfer for quantum approximate optimization of weighted maxcut," *ACM Transactions on Quantum Computing*, vol. 4, no. 3, pp. 1–15, 2023.

[74] G. S. Ravi, P. Gokhale, Y. Ding, W. Kirby, K. Smith, J. M. Baker, P. J. Love, H. Hoffmann, K. R. Brown, and F. T. Chong, "Cafqa: A classical simulation bootstrap for variational quantum algorithms," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, ser. ASPLOS 2023. New York, NY, USA: Association for Computing Machinery, 2022, p. 15–29. [Online]. Available: https://doi.org/10.1145/3567955.3567958

[75] M. Cain, E. Farhi, S. Gutmann, D. Ranard, and E. Tang, "The qaoa gets stuck starting from a good classical string," 2023.

[76] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, "An adaptive variational algorithm for exact molecular simulations on a quantum computer," *Nature communications*, vol. 10, no. 1, p. 3007, 2019.

[77] G. Gadzinski and A. Castello, "Combining white box models, black box machines and human interventions for interpretable decision strategies," *Judgment and Decision Making*, vol. 17, no. 3, pp. 598–627, 2022.

[78] A. Kudva, F. Sorourifar, and J. A. Paulson, "Constrained robust bayesian optimization of expensive noisy black-box functions with guaranteed regret bounds," *AIChE Journal*, vol. 68, no. 12, p. e17857, 2022.

[79] C. He, Y. Zhang, D. Gong, and X. Ji, "A review of surrogate-assisted evolutionary algorithms for expensive optimization problems," *Expert Systems with Applications*, p. 119495, 2023.