A Non-Variational Quantum Approach to the Job Shop Scheduling Problem

Miguel Angel Lopez-Ruiz[1], Emily L. Tucker[2], Emma M. Arnold[2], Evgeny Epifanovsky[1], Ananth Kaushik[1], and Martin Roetteler[1]

[1]IonQ Inc., 4505 Campus Dr, College Park, MD 20740, USA

[2]Department of Industrial Engineering, 211 Fernow St, Clemson University, Clemson, SC 29634-0920, USA

Quantum heuristics offer a potential advantage for combinatorial optimization but are constrained by near-term hardware limitations. We introduce Iterative-QAOA, a variant of QAOA designed to mitigate these constraints. The algorithm combines a non-variational, shallow-depth circuit approach using fixed-parameter schedules with an iterative warm-starting process. We benchmark the algorithm on Just-in-Time Job Shop Scheduling Problem (JIT-JSSP) instances on IonQ Forte Generation QPUs, representing some of the largest such problems ever executed on quantum hardware. We compare the performance of the algorithm against both the Variational Quantum Imaginary Time Evolution (VarQITE) algorithm and the non-variational Linear Ramp (LR) QAOA algorithm. We find that Iterative-QAOA robustly converges to find optimal solutions as well as high-quality, near-optimal solutions for all problem instances evaluated. We evaluate the algorithm on larger problem instances up to 97 qubits using tensor network simulations. The scaling behavior of the algorithm indicates potential for solving industrial-scale problems on fault-tolerant quantum computers.

1 Introduction

The Job Shop Scheduling Problem (JSSP) is among the most widely studied combinatorial optimization problems [1–3]. Its variants arise regularly in industry [4], most notably supporting advanced manufacturing [5–7]. In its baseline form, there are a set of J jobs that should be manufactured across a set of M machines in a sequence of operations. A common objective is to minimize makespan (i.e., the total length of time from start of production to completion) [3]. Alternative objectives (e.g., minimize mean flow time [8], energy [9], and workload [10]) allow companies to tailor to their particular setting. Its many variants include the Just-in-Time JSSP (JIT-JSSP) in which each task has a due date [11]; the Flexible JSSP (FJSSP) [12, 13] in which a task can be accomplished on multiple machines; and the dynamic JSSP in which jobs arrive after production begins [14].

JSSP is an NP-hard problem [15], and additional jobs increase the complexity of the problem exponentially [16]. Exact solution approaches have focused on branch-and-bound or disjunctive methods [17–19]. In practice, heuristics are commonly used [20] because of the problem complexity and computational time. These include the shifting bottleneck [17], local search [21], or simple dispatching rules (e.g., shortest processing time [7, 22]). Metaheuristic methods are also widespread (e.g., genetic algorithms [23, 24] and Tabu search [25, 26]).

While classical heuristics can yield high-quality solutions for the JSSP, its NP-hard nature motivates the exploration of alternative computational paradigms, such as quantum computing. Quantum approaches for solving the JSSP can be broadly categorized by their hardware paradigm. Quantum annealing has been used to solve QUBO formulations of both the standard JSSP [27–30] and the more complex FJSSP [31–33]. In gate-based quantum computing, most prior research relied on variational quantum algorithms (VQAs) [34–36]; however, non-variational

Miguel Angel Lopez-Ruiz: miguel.lopezruiz@ionq.co
Emily L. Tucker: etucke3@clemson.edu

approaches have also been explored, e.g., digitized adiabatic evolution in Ref. [37]. Within this paradigm, a significant focus has also been placed on developing more efficient encodings to reduce the resource requirements for near-term hardware [38–40].

Among the various gate-based methods, the Quantum Approximate Optimization Algorithm (QAOA) has emerged as a leading hybrid quantum-classical method for tackling combinatorial optimization problems [41]. Conceptually, QAOA can be viewed as a time-discretized form of adiabatic quantum computing. Its relatively simple circuit structure makes it well-suited for implementation on Noisy Intermediate-Scale Quantum (NISQ) devices, and there is evidence that, for certain problem classes, it may offer an advantage over classical algorithms [42–44]. Nevertheless, QAOA lacks general performance guarantees, and the classical optimization of its parameters can be a significant challenge [45].

A key determinant of performance in VQAs, such as QAOA, is the choice of initial state. A well-chosen starting point can reduce convergence time, help avoid poor local minima during the classical optimization stage, and improve solution quality. A promising strategy for enhancing QAOA is "warm-starting", in which a biased initial state is prepared using prior information rather than starting from a uniform superposition [46]. Moreover, beyond accelerating convergence, an appropriate initialization may also mitigate training issues such as barren plateaus [47].

Multiple warm-starting strategies have been proposed. One approach leverages other quantum processes, such as using the output of a quantum annealing run to prepare the initial QAOA state [48, 49]. Another employs iterative feedback, introducing adaptive longitudinal bias fields in the mixer Hamiltonian that are dynamically updated during the optimization process [50–52]. A prominent approach draws on classical heuristics, where an approximate classical solution is used to construct a biased initial state that steers the quantum search toward promising regions of the Hilbert space [46, 53]. This idea has been extended by combining classical warm-starts with engineered mixer Hamiltonians [54, 55]. Orthogonal to state initialization, other heuristics seek effective QAOA parameters bypassing the classical optimization stage, for instance through optimized or extrapolated parameter schedules from small problem instances [56–65].

In this work, we introduce a QAOA variant that combines a non-variational heuristic with an iterative warm-starting process. The algorithm leverages a fixed parameter schedule, bypassing the need for a classical optimization loop and any associated barren plateaus. The core of our method is a feedback mechanism that refines the initial state at each iteration by computing a bias-field using only information from previous quantum measurement outcomes [66]. A key distinction from prior work is our use of a normalized Boltzmann distribution to assign weights to measurement outcomes for the bias-field update. This technique inherently gives greater significance to results associated with lower energy states.

The remainder of this paper is structured as follows. In Section 2, we introduce the general mathematical formulation of the JSSP. Section 3 provides an overview of the general approach for mapping combinatorial optimization problems to quantum Hamiltonians, details the construction of our proposed algorithm, and introduces the variational algorithm used for comparison. In Section 4, we describe the specific JSSP problem formulation and present the results of applying the algorithms to various problem sizes in both ideal simulations and on quantum hardware. Finally, in Section 5, we summarize our findings and discuss avenues for future research.

2 Job Shop Scheduling Problem

The Job Shop Scheduling Problem (JSSP) [2] is a classical combinatorial optimization problem renowned for its computational complexity and broad industrial relevance. The problem involves scheduling a set of J jobs, denoted as $\mathcal{J} = \{j_1, \dots, j_J\}$, on a set of M machines, $\mathcal{M} = \{m_1, \dots, m_M\}$. Each job j consists of a sequence of operations that must be executed in a specific order. Each operation has a pre-assigned machine and a fixed processing time. The primary objective is to find a schedule that minimizes the makespan, i.e., the total time required to complete all jobs, subject to two main constraints: each machine can only process one operation at a time, and the precedence order of operations for each job must be respected.

A common approach to solve the JSSP on a quantum computer is to first map it to a Quadratic Unconstrained Binary Optimization (QUBO) problem. Following the time-indexed representation approach [29, 35], the schedule is defined using a set of binary variables x_{mjt}, where the indices represent the machine, job, and start time slot, respectively

$$x_{mjt} = \begin{cases} 1 & \text{if the operation of job } j \text{ on} \\ & \text{machine } m \text{ starts at time } t, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

This variable definition presumes that for any given job j, there is at most one operation that runs on a specific machine m, which holds for the standard JSSP.

The time index t is discretized and constrained by a deadline T, which represents the maximum allowable time for all jobs to complete. T is bounded and can be estimated heuristically. Let p_{mj} be the processing time of the operation for job j on machine m. The upper bound on the makespan, $T_{\max}$, is the total processing time of all operations

$$T_{\max} = \sum_{j\in\mathcal{J}} \sum_{m\in\mathcal{M}_j} p_{mj}, \tag{2}$$

where $\mathcal{M}_j$ is the set of machines that job j must visit. The lower bound $T_{\min}$ is the time required for the single longest job

$$T_{\min} = \max_{j\in\mathcal{J}} \sum_{m\in\mathcal{M}_j} p_{mj}. \tag{3}$$

The JSSP constraints are enforced by constructing a cost function $C(\mathbf{x})$ composed of penalty terms. A valid schedule must satisfy three primary constraints: (1) each operation must be scheduled exactly once (job assignment constraint); (2) no two operations can be processed on the same machine simultaneously (time assignment constraint); and (3) the original precedence order of operations within each job must be preserved (process order constraint). A schedule is feasible if and only if all of these penalty terms are zero. To guide the search towards solutions with small makespans, an objective term is added that penalizes late finishing times. The final cost function is a sum of these penalty and objective terms

$$C(\mathbf{x}) = \lambda_1 c_1(\mathbf{x}) + \lambda_2 c_2(\mathbf{x}) + \lambda_3 c_3(\mathbf{x}) + c_{\text{obj}}(\mathbf{x}), \tag{4}$$

where $\mathbf{x}$ is the complete set of binary variables and $\{\lambda_i\}$ are penalty weights chosen to be large enough to enforce the constraints. Each term $c_i(\mathbf{x})$ and $c_{\text{obj}}(\mathbf{x})$ is constructed to be at most quadratic in the variables x_{mjt}, thus ensuring the entire cost function is a valid QUBO [35].

3 Quantum Approach to Combinatorial Optimization

Having formulated the JSSP as a QUBO problem, the goal is to find the binary assignment $\mathbf{x}^\star$ that minimizes the cost function in Eq. (4). In the quantum computing paradigm, this is achieved by finding the ground state of a corresponding Ising Hamiltonian. To map the classical QUBO cost function $C(\mathbf{x})$ to a quantum Hamiltonian H_C, each binary variable x_k in the vector $\mathbf{x}$ is promoted to a quantum operator acting on a single qubit via the transformation $x_k \to (1-\sigma_z^k)/2$, where σ_z^k is the Pauli-Z matrix acting on the k-th qubit. This converts the cost function into a diagonal Hamiltonian of the form

$$H_C = C\left(x_k \to \frac{1-\sigma_z^k}{2}\right). \tag{5}$$

In this representation, the optimal solution to the JSSP corresponds to the ground state of H_C.

The predominant strategy for finding this ground state on current generation of NISQ hardware has been through VQAs. In a VQA, one prepares a parameterized trial state (ansatz) $|\psi(\boldsymbol{\theta})\rangle$, on the quantum computer. A classical optimizer then iteratively updates the real-valued parameters $\boldsymbol{\theta}$ to minimize the expectation value of the Hamiltonian, $\langle\psi(\boldsymbol{\theta})|H_C|\psi(\boldsymbol{\theta})\rangle$. This iterative quantum-classical optimization loop characteristic of VQAs can suffer from challenges such as “barren plateaus”, which are regions in the parameter landscape where cost function gradients with respect to the variational parameters vanish exponentially with increasing problem size [45], making the classical optimization intractable. Furthermore, performance is highly dependent on the choice of the ansatz, and the optimization can become trapped in poor local minima.

To address these challenges, in this work we introduce the Iterative-QAOA, a non-variational variant of the standard QAOA, and benchmark its performance against the Variational Quantum Imaginary Time Evolution (VarQITE) for solving the JSSP.

3.1 Iterative-QAOA

The Iterative-QAOA algorithm is built upon the standard QAOA [41, 42] framework, which we briefly summarize here. The algorithm begins by encoding the problem into a cost Hamiltonian H_C in Eq. (5), whose ground state corresponds to the optimal solution. A parameterized quantum state, or ansatz, is then prepared on a quantum computer. Starting from an initial state $|\psi_{\text{init}}\rangle$, typically the uniform superposition $|+\rangle^{\otimes N}$, the p-layer QAOA ansatz is constructed by alternately applying unitary operators derived from the cost Hamiltonian H_C and a mixer Hamiltonian H_M. A common choice for the mixer, is the transverse field Hamiltonian, $H_M = \sum_{i=1}^{N} \sigma_x^i$. The full ansatz is then given by

$$|\psi_p(\boldsymbol{\gamma}, \boldsymbol{\beta})\rangle = \prod_{k=1}^{p} e^{-i\beta_k H_M} e^{-i\gamma_k H_C} |\psi_{\text{init}}\rangle, \qquad (6)$$

where $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ are $2p$ real-valued variational parameters. A classical optimizer is then used to find the optimal parameters $(\boldsymbol{\gamma}^\star, \boldsymbol{\beta}^\star)$ by minimizing the expectation value of the cost Hamiltonian H_C. However, as mentioned earlier, this quantum-classical process can be resource-intensive and is susceptible to challenges common to VQAs.

To circumvent this often costly classical optimization, several non-variational QAOA strategies have been proposed. In these approaches, the parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ are determined by a predefined schedule rather than being variationally optimized. Such schedules are frequently inspired by insights from quantum annealing and adiabatic evolution [57–59], where the QAOA procedure is viewed as a digitized version of an analog process [48]; or by exploiting symmetries of the classical objective function [60]. A prime example and arguably the most straightforward schedule is used in the Linear Ramp QAOA (LR-QAOA) [61–63]. The LR-QAOA protocol replaces the variational search for optimal angles with a fixed, linear schedule. For a p-layer ansatz, this schedule is defined by only two global hyperparameters, Δ_γ and Δ_β, which set the range for the respective parameter ramps. The parameters for layer k (for $k = 0, \ldots, p-1$) are then given by

$$\beta_k = \left(1 - \frac{k}{p}\right) \Delta_\beta, \qquad \gamma_k = \frac{k+1}{p} \Delta_\gamma. \qquad (7)$$

To enhance the performance of fixed-schedule protocols such as LR-QAOA, we propose an iterative method that refines the algorithm's initial state. Our approach (Iterative-QAOA) periodically updates the starting state based only on measurement outcomes from the previous run [66], while keeping the number of QAOA layers and the parameter schedule fixed. After each run j, the set of measurement bitstrings $\{s_j\}$ and their corresponding energies $\{E_j\}$, where $E_j = \langle s_j | H_C | s_j \rangle$, is used to guide the search for the next iteration.

The update mechanism calculates a bias [67] for each qubit q by computing the thermal expectation value of its Pauli-Z operator σ_z^q, from the measurement outcomes. This is achieved using a Boltzmann distribution over the measured states, $P(s_j) = \exp(-\beta_T E_j)/Z$, where $Z = \sum_k \exp(-\beta_T E_k)$ is the canonical partition that normalizes the distribution and β_T is a hyperparameter, analogous to the inverse temperature of the ensemble. The role of β_T here is to control the intensity of the feedback; a large value creates a sharp probability distribution that focuses the update on the lowest-energy states, while a smaller value incorporates a wider sample of outcomes for a more gradual update. The bias for each qubit is then the thermal expectation value of σ_z^q with respect to this distribution

$$\langle \sigma_z^q \rangle_T = \sum_i P(s_i) \langle s_i | \sigma_z^q | s_i \rangle. \qquad (8)$$

From these biases, a new initial product state $|\psi_{\text{init}}^{(j+1)}\rangle$ is prepared for the next iteration $(j+1)$. The state for each qubit is updated based on the probability p_q of it being initialized to the $|1\rangle$ state, calculated as

$$p_q = \frac{1}{2}\left(1 - \eta \langle \sigma_z^q \rangle_T\right), \qquad (9)$$

where $\eta \in \{-1, 1\}$ is a hyperparameter that directs the feedback, either reinforcing $(\eta = +1)$ or reversing $(\eta = -1)$ the trend observed in the low-energy states. The new initial state is a product state of the form

$$|\psi_{\text{init}}^{(j+1)}\rangle = \bigotimes_{q=1}^{N} \left(\sqrt{1 - p_q}|0\rangle_q + \sqrt{p_q}|1\rangle_q\right). \qquad (10)$$

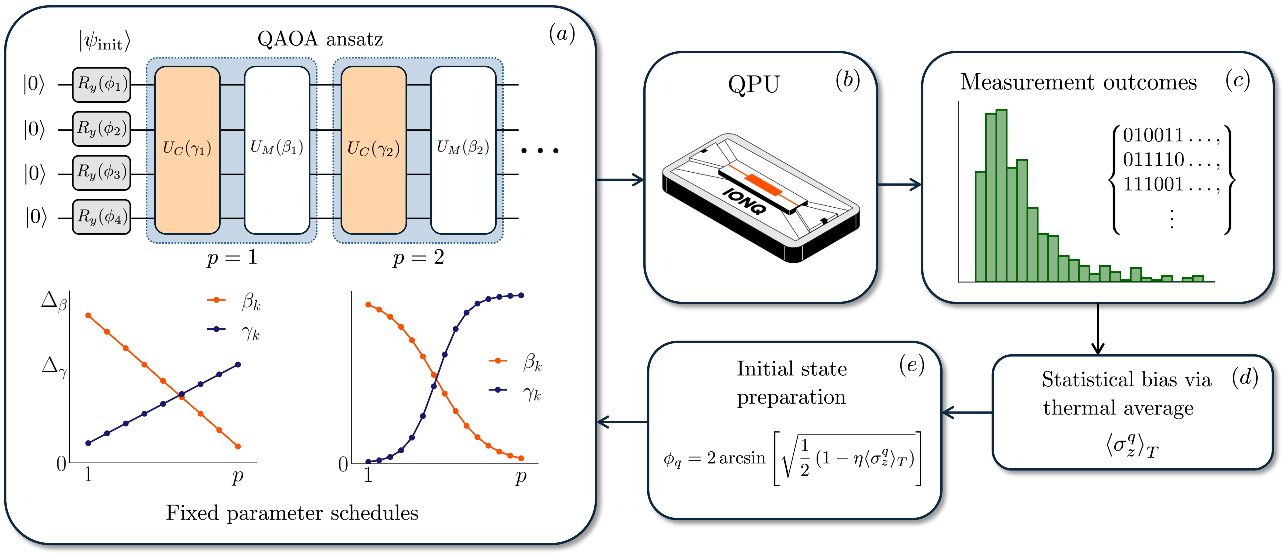

Figure 1: Workflow of the Iterative-QAOA algorithm. (a) A non-variational QAOA ansatz is prepared using a fixed number of layers p and a predetermined schedule for the angles $\{\beta_k\}$ and $\{\gamma_k\}$. The circuit applies alternating cost $(U_C(\gamma_k) = \exp[-i\gamma_k H_C])$ and mixer $(U_M(\beta_k) = \exp[-i\beta_k H_M])$ unitaries to an initial state $|\psi^{(j)}_{\text{init}}\rangle$. (b) This circuit is executed on quantum hardware (QPU) yielding (c) a set of classical bitstrings. (d) These measurement outcomes are used to compute a statistical bias for each qubit, given by the thermal expectation value $\langle\sigma_z^q\rangle_T$. (e) The computed bias determines the angles $\{\phi_q\}$ of the next iteration's initial state $|\psi^{(j+1)}_{\text{init}}\rangle$, which is prepared by a layer of single-qubit rotations $R_y(\phi_q)$.

This state is efficiently prepared by applying a layer of single-qubit rotations $R_y(\phi_q)$ to the $|0\rangle^{\otimes N}$ state, with rotation angles given by $\phi_q = 2\arcsin\left(\sqrt{p_q}\right)$ and the mixer Hamiltonian H_M is suitably modified such that this new initial state is an eigenstate of the mixer [46]. The algorithm proceeds by repeating this process which progressively directs the search towards areas of the Hilbert space corresponding to optimal solutions. In Fig. 1 we summarize the workflow of this algorithm.

3.2 Variational Quantum Imaginary Time Evolution

The Variational Imaginary Time Evolution (VarQITE) algorithm provides an efficient method to approximate the ground state $|\psi_{\text{GS}}\rangle$ of a Hamiltonian H_C, such as those encoding combinatorial optimization problems. In imaginary time $\tau \equiv it$, the normalized quantum state evolves according to

$$\frac{\partial}{\partial\tau}|\psi(\tau)\rangle = -(H_C - E_\tau)|\psi(\tau)\rangle, \tag{11}$$

where E_τ is a normalization factor. For a static Hamiltonian, the formal solution

$$|\psi(\tau)\rangle = \frac{e^{-\tau H_C}}{\sqrt{\langle\psi(0)|e^{-2\tau H_C}|\psi(0)\rangle}}|\psi(0)\rangle \tag{12}$$

projects any initial state, with non-zero ground-state overlap, toward $|\psi_{\text{GS}}\rangle$ as $\tau \to \infty$.

Implementing the non-unitary operator $\exp(-\tau H_C)$ directly on quantum hardware is, however, impractical. Instead, VarQITE approximates this evolution using a parameterized quantum circuit $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|+\rangle^{\otimes N}$, where $U(\boldsymbol{\theta})$ defines a unitary ansatz characterized by real parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_{N_p})$. The goal is to update these parameters as functions of imaginary time, $\boldsymbol{\theta}(\tau)$, so that the variational state $|\psi(\boldsymbol{\theta}(\tau))\rangle$ approximates the exact imaginary-time-evolved state. Rather than using the McLachlan variational principle [68, 69], which requires $\mathcal{O}(N_p^2)$ circuit evaluations per step, we employ the more efficient formulation of Ref. [70], which enforces the dynamics of each Pauli term P_i in H_C through

$$\begin{aligned}\frac{\partial\langle P_i\rangle}{\partial\tau} &= \sum_j 2\,\mathrm{Re}\left[\langle\psi(\boldsymbol{\theta})|\,P_i\,\frac{\partial}{\partial\theta_j}|\psi(\boldsymbol{\theta})\rangle\right]\dot{\theta}_j \\ &= -\langle\psi(\boldsymbol{\theta})|\{P_i, H_C - E_\tau\}|\psi(\boldsymbol{\theta})\rangle, \end{aligned}\tag{13}$$

yielding the linear system of the form $\sum_j A_{ij}\dot{\theta}_j = B_i$, where

$$A_{ij} = \mathrm{Re}\left[\langle\psi(\boldsymbol{\theta})|\, P_i \frac{\partial}{\partial\theta_j} |\psi(\boldsymbol{\theta})\rangle\right],$$
$$B_i = -\frac{1}{2}\langle\psi(\boldsymbol{\theta})|\{P_i, H_C - E_\tau\}|\psi(\boldsymbol{\theta})\rangle\,. \quad (14)$$

Each column of A is obtained from two parameter-shift measurements [71, 72], while for QUBO Hamiltonians containing only commuting Pauli-Z terms, all elements of B can be evaluated from a single circuit. The parameters are then evolved in imaginary time via a first-order update rule, $\boldsymbol{\theta} \to \boldsymbol{\theta} + \Delta\tau\dot{\boldsymbol{\theta}}$, where $\dot{\boldsymbol{\theta}}$ is computed by solving the above linear system.

This formulation requires only $2N_p + 1$ circuit evaluations per time step, significantly improving efficiency over previous implementations while retaining the key physics of imaginary-time evolution. It thus offers a practical and potentially scalable approach for ground-state preparation on near-term quantum hardware.

The imaginary time step $\Delta\tau$ is a critical hyperparameter governing the algorithm's convergence and accuracy. Its selection involves a trade-off between fidelity and efficiency; a small step size improves the accuracy of the Euler integration at the cost of more iterations, whereas a large step size can lead to numerical instability and hinder convergence. While a constant step size is the simplest approach, we find that using a step-size schedule $\Delta\tau(\tau)$ yields substantially better performance. This suggests that adaptive strategies, where $\Delta\tau$ is adjusted dynamically based on the system's state, could further enhance the algorithm's efficiency and robustness.

4 Results and Discussion

In this section, we present a comprehensive evaluation of the Iterative-QAOA algorithm applied to instances of a specific JSSP variant. We evaluate its performance against two key reference algorithms: the underlying non-variational LR-QAOA establishing the fixed parameter schedule, and VarQITE as a representative of the variational algorithms class. We begin by detailing the specific JSSP cost function used. The algorithms are then evaluated across a range of problem sizes (see Appendix A for details on problem instances) using both ideal noiseless simulations and executions on a trapped-ion quantum processor (see Appendix E for technical details of the quantum hardware) to assess the algorithms' performance and robustness to hardware noise. We conclude with an investigation of the method's scaling potential using large-scale simulations up to ~ 100 qubits.

4.1 JSSP Formulation

Following the formulation proposed in Ref. [36], we consider a JIT-JSSP variant tailored to a steel manufacturing process in which due dates are defined for each job, and the objective is to minimize deviations from the due dates and changeover (switching) costs.

In this problem, a set of J jobs must each be processed on M machines in a fixed order with fixed idle slots. The schedule of each machine is divided into T consecutive time slots, and each non-idle time slot must be assigned to exactly one job. The processing times for all job operations are assumed to be equal. To represent the schedule, we introduce binary decision variables $x_{mjt} \in \{0, 1\}$ that indicate whether job j is processed by machine m at time t.

Each job j has a due time d_j, and finishing before or after this time incurs a penalty that scales linearly with the deviation. The early/late delivery cost for job j can be formulated as follows

$$\begin{aligned} u_j(\mathbf{x}) = c_e &\sum_{t=1}^{d_j}(d_j - t)x_{Mjt} \\ &+ c_l \sum_{t=d_j+1}^{T}(t - d_j)x_{Mjt} \\ &\forall j = 1, ..., J, \end{aligned} \quad (15)$$

where c_e and c_l are constants controlling the magnitude of the early and late delivery penalties, respectively.

Each operation is associated with a production group, and switching production groups in consecutive time slots on the same machine incurs a switching cost. Let P_{mj} denote the production group of job j on machine m, and define $G^{(m)}_{j_1 j_2} = 1$ if $P_{mj_1} \neq P_{mj_2}$ and 0 otherwise. The production switching cost for machine m is then given by

$$s_m(\mathbf{x}) = c_p \sum_{j_1,j_2=1}^{J} \sum_{t=1}^{T_m-1} G^{(m)}_{j_1 j_2} x_{mj_1 t} x_{mj_2(t+1)} \quad \forall m = 1,\dots,M, \tag{16}$$

where c_p is the production switching cost coefficient. The total cost for a given schedule is then

$$c(\mathbf{x}) = \sum_{j=1}^{J} u_j(\mathbf{x}) + \sum_{m=1}^{M} s_m(\mathbf{x}). \tag{17}$$

The schedule of each machine comprises a tight sequence of J jobs within the total number of time slots T_m, on machine m, and feasibility is ensured by the following three sets of constraints. *Job assignment constraints* ensure that each job j is scheduled exactly once on each machine m:

$$g_{mj}(\mathbf{x}) = \sum_{t=1}^{T_m} x_{mjt} = 1 \quad \forall m = 1,\dots,M;\ j = 1,\dots,J. \tag{18}$$

The *time assignment constraints* limit each time slot t on machine m to its appropriate capacity. Let

$$\ell_{mt}(\mathbf{x}) = \sum_{j=1}^{J} x_{mjt} \quad \forall m = 1,\dots,M;\ t = 1,\dots,T_m, \tag{19}$$

and let A_m and I_m denote, respectively, the sets of active and idle slots on machine m, with $A_m \cup I_m = \{1,\dots,T_m\}$ and $A_m \cap I_m = \emptyset$. We impose

$$\ell_{mt}(\mathbf{x}) = \begin{cases} 1, & \forall m = 1,\dots,M;\ t \in A_m, \\ 0, & \forall m = 1,\dots,M;\ t \in I_m. \end{cases} \tag{20}$$

Finally, the *process order constraints* enforce that jobs must progress through the machines in the correct order, preventing any job from appearing earlier on a later machine or appearing on multiple machines in the same time slot:

$$q_{mj}(\mathbf{x}) = \sum_{t=2}^{J} \sum_{t'=1}^{t} x_{mjt} x_{(m+1)jt'} = 0, \quad \forall m = 1,\dots,M-1;\ j = 1,\dots,J. \tag{21}$$

To incorporate these constraints within a quantum framework, we convert the total cost function into a QUBO form by adding penalty terms yielding

$$\begin{aligned} C(\mathbf{x}) = c(\mathbf{x}) &+ \lambda \sum_{m=1}^{M} \sum_{j=1}^{J} (g_{mj}(\mathbf{x}) - 1)^2 \\ &+ \lambda \sum_{m=1}^{M} \left[\sum_{t\in A_m} (\ell_{mt}(\mathbf{x}) - 1)^2 + \sum_{t\in I_m} \ell_{mt}(\mathbf{x})^2 \right] \\ &+ \lambda \sum_{m=1}^{M-1} \sum_{j=1}^{J} q_{mj}(\mathbf{x}), \end{aligned} \tag{22}$$

where $\lambda > 0$ is a penalty weight. In this formulation, the idle slots are treated as fixed zero-capacity positions.

This formulation closely follows that of Ref. [36], with only two key modifications. First, we enforce a stricter sequential processing constraint, requiring a job to finish on one machine before starting on the next. This is reflected in the upper limit of the inner sum in the process order constraint (Eq. (21)), which changes from $t-1$ to t. Second, we treat idle slots (I_m) as fixed parameters in the problem definition, rather than handling them as optimizable dummy variables as is done in Ref. [36]. This modification simplifies the model by reducing the total number of variables.

It is worth noting that this setup defines due dates at the job level. More complex variants of the JIT-JSSP exist where due dates apply at the operation level, leading to significantly higher complexity and greater difficulty in obtaining optimal solutions [11].

The JIT-JSSP instances evaluated in this work originate from a master problem comprising $J = 20$ jobs and $M = 3$ machines. Using the time-indexed formulation with binary variables x_{mjt} (Eq. (1)), the total number of variables for the full instance is $n_{\text{var}} = \sum_{m=1}^{M} JT_m = 1,300$, where T_m denotes the number of time slots per machine m, which in this case is $T_m = \{20, 22, 23\}$ for $m \in \{1, 2, 3\}$ respectively. In our quantum approach, a one-hot encoding (Eq. (5)) is utilized to map each binary variable to a qubit. Solving the full 1,300-variable instance via quantum simulation or on current quantum hardware is intractable. However, state-of-the-art classical solvers are capable of handling instances of this size (see Appendix B). As detailed in Appendix A, the full instance was solved classically, yielding an optimal makespan of 193. The sub-instances were then constructed by fixing ("freez-

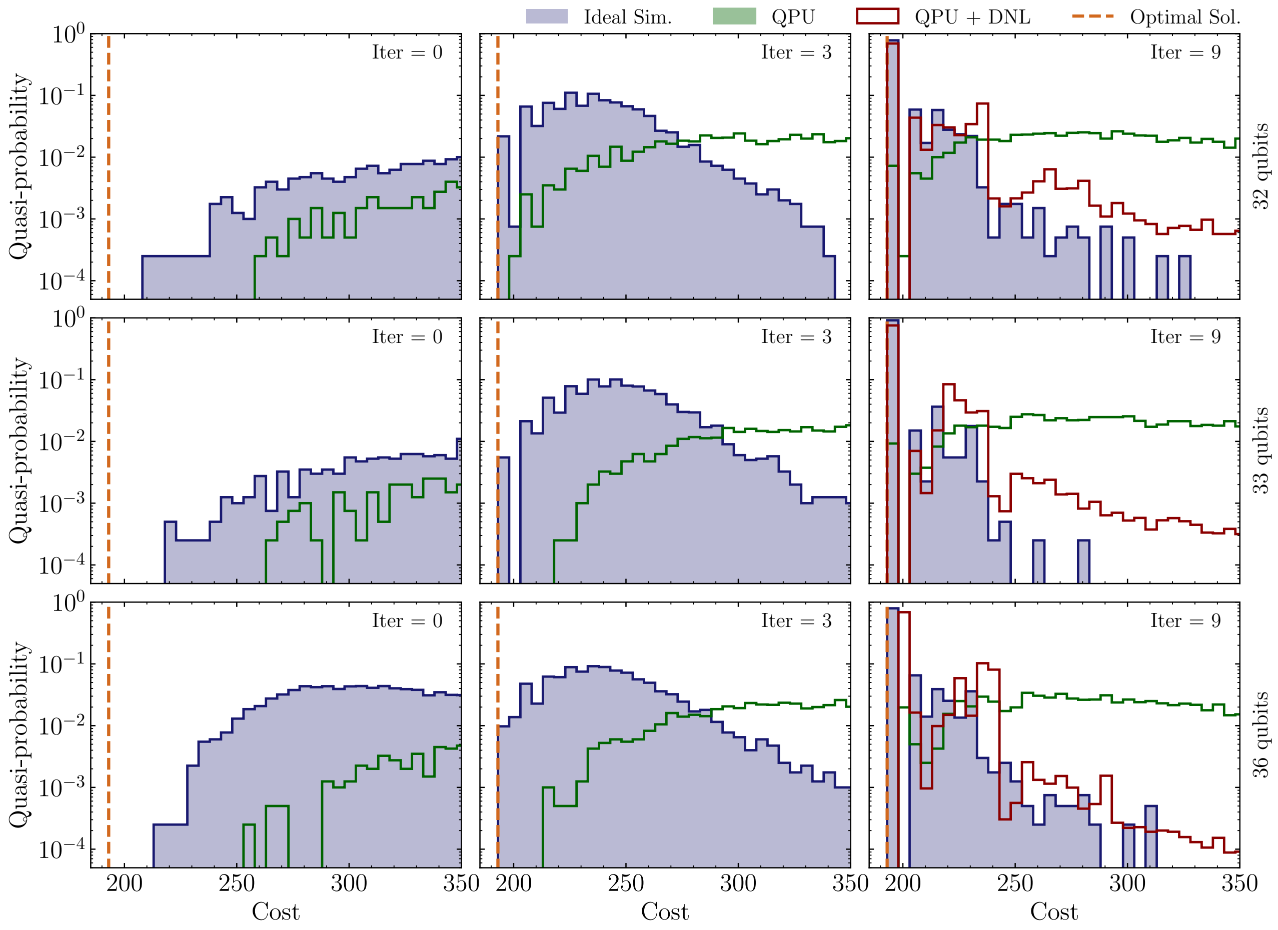

Figure 2: Performance of Iterative-QAOA on JSSP instances of two different sizes executed on IonQ Forte Generation QPU. Shown here are the low-energy sectors of the cost probability distributions for the 32- (top row), 33- (middle row), and 36- (bottom row) qubit instances. The columns display the distribution's evolution at the initial (Iter = 0), an intermediate (Iter = 3), and the final (Iter = 9) iteration. Each panel compares the ideal noiseless simulation results with raw data from the QPU executions and the error-mitigated results after applying DNL. The dashed vertical line indicates the known optimal cost. The algorithm parameters used are $\Delta_{\beta/\gamma} = 0.17$ and a quadratic schedule for $\beta_T \in [0.1, 1]$.

ing") most variables to the optimal solution, allowing us to generate problems ranging from 24 to 36 qubits for QPU execution, and larger 50- and 97-qubit instances suitable for tensor network simulations. Hence, the optimal schedule for all sub-problems corresponds to the classically computed global optimum, with a ground state energy of 193. The detailed methodology for generating these sub-instances is described in Appendix A, where we also show the specific problem parameters (Tables 2 and 3).

4.2 Performance of Iterative-QAOA

We evaluate the performance of our Iterative-QAOA algorithm on JIT-JSSP instances with 32, 33, and 36 qubits (see Fig. 9). The quantum circuit within each iteration is based on the LR-QAOA schedule (see Eq. (7)), with an ansatz depth of $p = 5$ for the 32- and 33-qubit instances and $p = 6$ for the 36-qubit instance. Based on an extensive parameter search detailed in Appendix C, an optimal ramp parameter of $\Delta_{\beta/\gamma} = 0.17$ was used for all instances. For the feedback loop that creates the updated initial state at each iteration, the inverse temperature hyperparameter β_T was set to a quadratic schedule that grows from 0.1 to 1.0 over 10 iterations. All runs were executed with 4,000 measurement shots per circuit both on the ideal quantum simulator as well as on quantum hardware. The two smallest instances were simulated using a statevector simulator while the ideal results for the 36-qubit instance were obtained using a Matrix

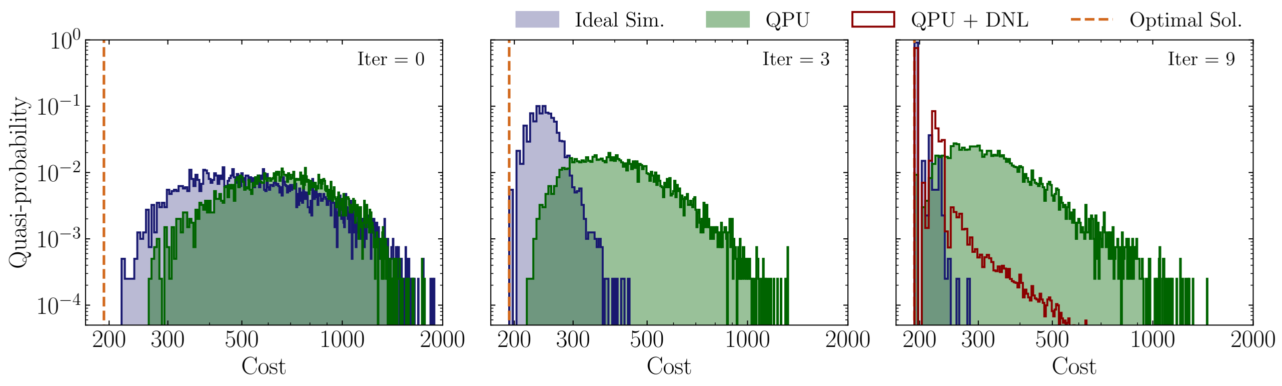

Figure 3: Evolution of the full cost distribution for the 33-qubit JSSP instance under Iterative-QAOA on the IonQ Forte Generation QPU. The panels display the distributions at the initial (Iter = 0), an intermediate (Iter = 3), and the final (Iter = 9) iteration. The algorithm parameters used are $\Delta_{\beta/\gamma} = 0.17$ and a quadratic schedule for $\beta_T \in [0.1, 1]$.

Product State (MPS) [73, 74] simulator with a maximum bond dimension of $\chi = 256$.

The evolution of the algorithm's performance is shown in Fig. 2, which displays the low-energy sector of the cost distribution at initial, intermediate, and final iterations. In ideal noiseless simulations, the algorithm proved to be highly effective. For all three problem instances, the iterative process converged successfully, producing a final distribution where the probability of sampling the optimal solution (i.e. the ground state) exceeded 78%.

On the QPU, the effects of hardware noise become more apparent as the algorithm evolves. This is further illustrated in Fig. 3, which shows the full cost distributions of the 33-qubit instance. In the ideal case, the algorithm rapidly "squeezes" the probability distribution towards the ground state, resulting in a final distribution highly skewed towards the optimal solution. The evolution on the hardware is qualitatively similar but less pronounced, indicating that noise effectively broadens the final distribution reducing the probability of sampling the ground state.

To mitigate the impact of hardware induced errors on the QPU executions, we applied a debiasing with non-linear filtering (DNL) post-processing technique to the raw measurement data (see Appendix F for more details). In both Figs. 2 and 3, the DNL technique effectively suppresses the high-energy tails of the noisy distribution while enhancing the probability of the target low-energy states. Nevertheless, even in the raw, unmitigated case, the algorithm demonstrated sufficient robustness to find the optimal solution for the 32- and 33-qubit instances and a high-quality sub-optimal solution, corresponding to the first excited state, for the 36-qubit problem.

To further probe the robustness of the algorithm, we repeated the experiments using a deliberately sub-optimal parameter configuration. We selected a ramp parameter of $\Delta_{\beta/\gamma} = 1.25$, a value far from the optimal region (see Appendix C), and a fixed $\beta_T = 0.5$. As detailed in Appendix D, we observed that the iterative warm-starting process was powerful enough to guide the less effective QAOA ansatz toward the optimal solution in both ideal simulations and hardware runs.

Moreover, we compared the performance of Iterative-QAOA against the underlying LR-QAOA. For this comparison, we used a 24-qubit JIT-JSSP instance with an optimized ramp parameter $\Delta_{\beta/\gamma} = 0.17$ (see Appendix C). The noiseless simulations were performed using 4,000 measurement shots.

The results are summarized in Fig. 4. For LR-QAOA, a shallow circuit with $p = 4$ layers did not sample the optimal solution. Increasing the depth to $p = 25$ and $p = 50$ resulted in ground state populations of 0.35% and 3.82%, respectively. In contrast, after only 10 iterations, the Iterative-QAOA protocol using a fixed depth of just $p = 4$ layers achieved a ground state sampling probability of approximately 97.92%.

This performance difference highlights the advantages of an efficient warm-starting process, particularly for current and near-term quantum hardware where errors accumulate rapidly with

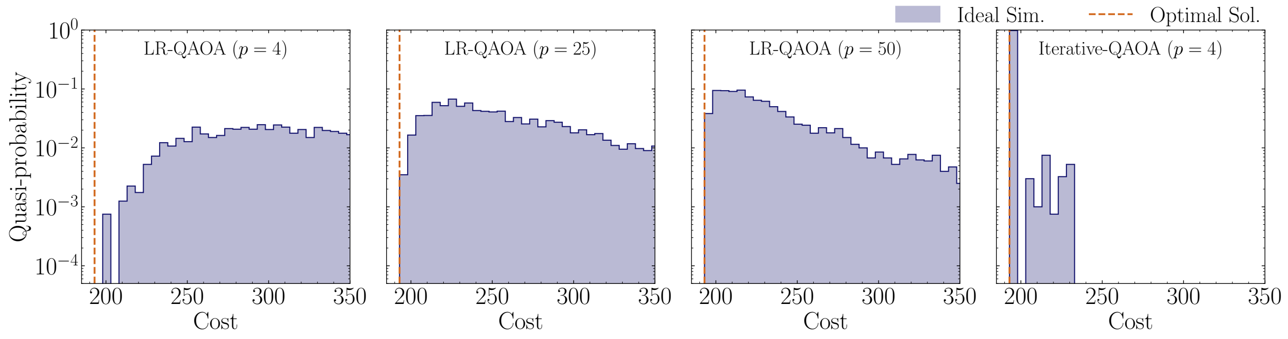

Figure 4: Comparison of LR-QAOA and Iterative-QAOA performance on the 24-qubit JIT-JSSP instance. All panels show the low-energy sector of the final cost distribution from ideal noiseless simulations with 4,000 measurement shots. The first three panels (from left to right) display the results for a standard LR-QAOA circuit with increasing depth of $p = 4, 25,$ and 50 layers, respectively. The results show that even at a significant depth of $p = 50$, the probability of sampling the ground state remains low ($< 4\%$). The rightmost panel shows the distribution for $p = 4$ Iterative-QAOA after 10 iterations, resulting in a ground state population of 97.92%. The algorithm parameters used are $\Delta_{\beta/\gamma} = 0.17$ and a quadratic schedule for $\beta_T \in [0.1, 1]$.

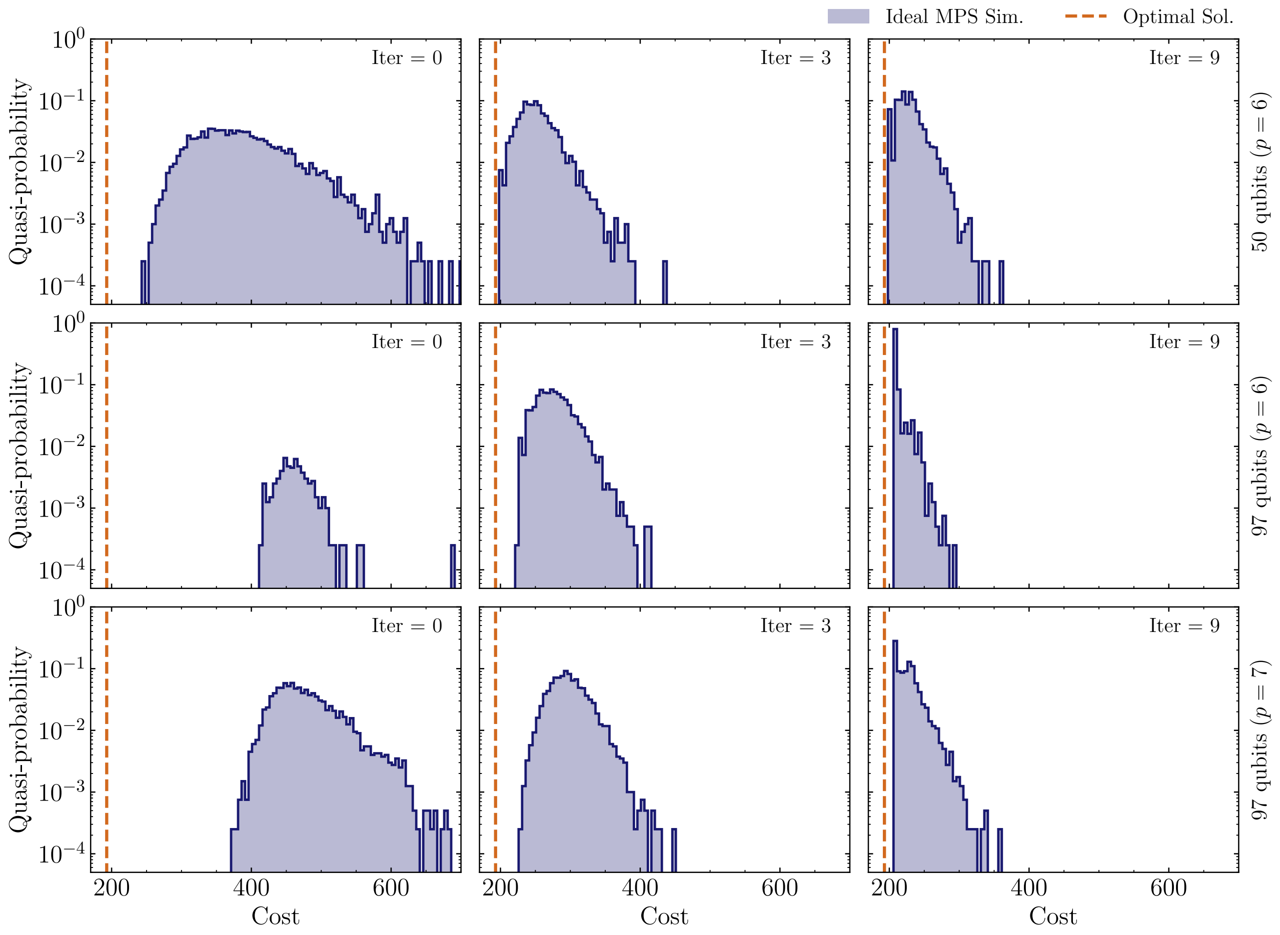

Figure 5: Performance of Iterative-QAOA on 50- and 97-qubit JIT-JSSP instances with an ideal MPS simulator and bond dimension $\chi = 256$. Each panel shows the low-energy sectors of the cost probability distributions at the initial (Iter = 0), an intermediate (Iter = 3), and the final (Iter = 9) iteration. The number of layers for the 50-qubit instance was $p = 6$ while for the 97-qubit instance we show the two cases of $p = 6$ and 7. The algorithm parameters used are $\Delta_{\beta/\gamma} = 0.17$ and a quadratic schedule for $\beta_T \in [0.1, 1]$.

circuit depth. The $p = 50$ LR-QAOA circuit requires 4,824 single-qubit and 3,900 two-qubit gates. The shallow $p = 4$ circuit used in each Iterative-QAOA run, however, requires only 408 single-qubit and 312 two-qubit gates; an order of magnitude fewer resources per iteration. This makes the iterative approach significantly more robust and practical for execution on current devices.

Furthermore, the scaling of LR-QAOA presents a challenge for larger problem instances. As detailed in Appendix C, the circuit depth required for LR-QAOA to achieve a certain solution quality is expected to grow with the problem size, reaching prohibitive gate counts, particularly for larger instances on near term quantum computers.

Finally, to probe the algorithm's scaling on instances beyond the capacity of the quantum hardware, we simulated 50- and 97-qubit JIT-JSSP instances. Given the large Hilbert space, as with the 36-qubit instance, these simulations were performed using an MPS simulator with a fixed, maximum bond dimension of $\chi = 256$. We used an ansatz depth of $p = 6$ for the 50-qubit case, and explored both $p = 6$ and $p = 7$ for the 97-qubit case, with 4,000 measurement shots for all runs. To test the robustness of our hyperparameter selection, we used the same values of $\Delta_{\beta/\gamma} = 0.17$ and a quadratic schedule for β_T.

The results, shown in Fig. 5, indicate that both large-scale instances demonstrated convergence after 10 iterations, with their final probability distributions becoming heavily skewed towards the low-energy sector. Although the true ground state was not sampled in either case, the 50-qubit instance sampled the first excited state of the Hamiltonian. In contrast, the 97-qubit runs converged to a solution of slightly lower quality (4th excited state), indicating increased difficulty of the problem at this scale. One potential reason for this decreased solution quality could be that the circuit depths of $p = 6$ and $p = 7$ may not be sufficiently expressive to resolve the ground state for problems of this scale. Another possible reason could be that increasing circuit depth also generates more entanglement, requiring a larger bond dimension to maintain accuracy in the MPS approximation, and thus greater classical computational resources. This effect is visible in the 97-qubit case, where increasing the depth from $p = 6$ to $p = 7$, yielded no improvement in the final energy. This suggests that, at a fixed bond dimension, the simulation could not capture all additional correlations generated by the deeper circuit. Nevertheless, the ability of Iterative-QAOA to consistently find high-quality sub-optimal solutions, even within this constrained scenario, remains a strong indicator of its potential for large-scale problems. All the results discussed in this section are summarized in Table 1.

4.3 Performance of VarQITE

We also evaluated the performance of the VarQITE algorithm on a representative JIT-JSSP instance of 32 variables, corresponding to a 32-qubit Hamiltonian. For the variational circuit, we employ a two-layer ansatz with nearest-neighbor connectivity and $N_p = 64$ variational parameters, as depicted in Fig. 6. As pointed out in Section 3.2, a critical hyperparameter for VarQITE is the imaginary time step $\Delta\tau$. Through preliminary optimization, we identified a variable step-size schedule of the form

$$\Delta\tau(\tau) = \Delta_0(1 - r)^\tau, \tag{23}$$

to be most effective, using the optimized values $\Delta_0 = 0.1$ and $r = 0.06$ for the evolution.

The algorithm was executed for 65 imaginary time steps, at which point convergence was observed. We performed both ideal noiseless simulations and executions on quantum hardware, using 120 measurement shots for each of the 129 circuits required per time step. The results of these runs are portrayed in Fig. 7 and summarized in Table 1. The best schedule sampled from the ideal simulation corresponded to a cost of 228, while the QPU execution yielded a lower best cost of 208. In both cases, this best solution was found with a low probability of approximately 0.83% (corresponding to a single measurement shot). Relative to the optimal value of 193, these costs correspond to the fourteenth and fourth excited states of the problem Hamiltonian, respectively.

Although VarQITE did not reach the ground state for this 32-qubit problem, the results highlight two of its key characteristics. First, the algorithm exhibits a surprising robustness to hardware noise. The overall convergence on the QPU was comparable to the ideal simulation, yet it sampled a single state with a lower cost. Given that this occurred in only a single measurement shot, it is likely due to statistical fluctuation induced by noise. Second, in smaller-scale test runs,

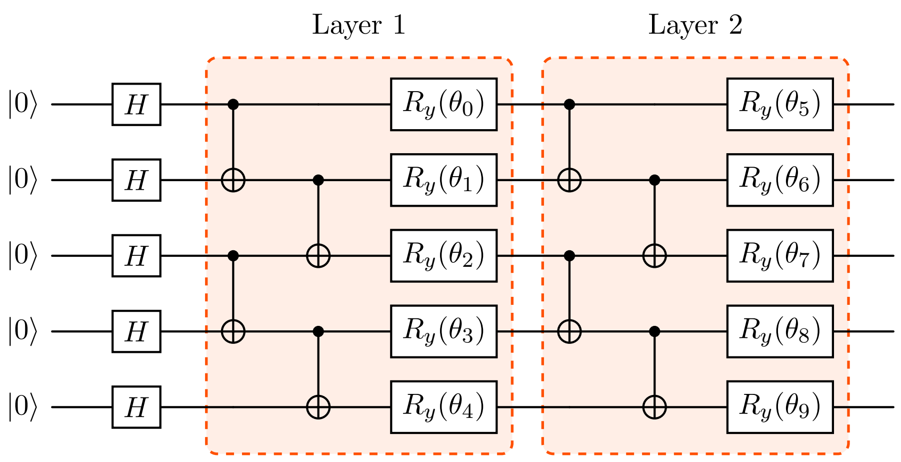

Figure 6: Structure of the two-layer variational ansatz $|\Psi(\boldsymbol{\theta})\rangle$ used for VarQITE runs. The figure shows a simplified 5-qubit version to illustrate the overall layout. The full 32-qubit circuit consists of $N_p = 64$ variational parameters.

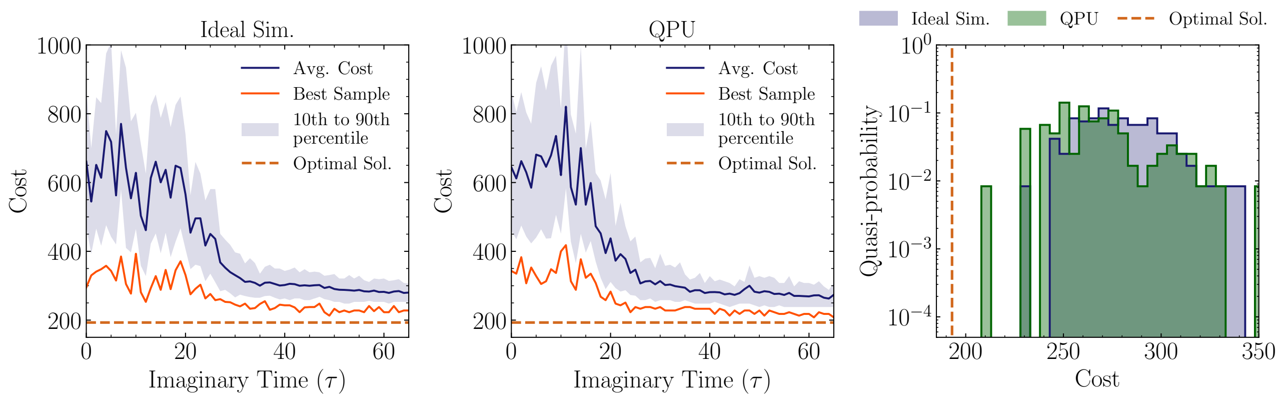

Figure 7: Performance of the VarQITE algorithm on the 32-qubit JIT-JSSP instance. The leftmost plots show the convergence for the ideal noiseless simulation and the QPU execution, respectively, as a function of imaginary time τ. In both convergence plots, the solid blue line corresponds to the average cost $\langle H_C \rangle$, while the shaded region indicates the range between the 10th and 90th percentiles of the sampled distribution. The orange line tracks the cost of the best (lowest-energy) sampled at each time step. The right-most plot displays the low-energy sector of the cost distribution of the last time step.

the algorithm performed significantly better indicating potential challenges for scaling this algorithm to larger problem instances.

4.4 Discussion

The key results presented in the preceding sections are summarized in Table 1. The classical computational time required to solve these instances scales approximately exponentially with the number of variables, as shown in Fig. 10 of Appendix B. Given that the number of variables scales at least quadratically with the number of jobs J (specifically as $O(J^2M)$ when $T_m \approx J$), it follows that even moderate increases in problem size could potentially render the problem classically impractical. For instance, extrapolating the observed scaling suggests that a problem with $J = 30$ jobs and $M = 3$ machines (corresponding to at least 2,700 binary variables) would require approximately 4 days to solve using the benchmarked classical solver (see Fig. 10).

Investigating the potential quantum scaling is therefore crucial. Our exploration using 50- and 97-qubit MPS simulations provides preliminary insights, albeit with limitations inherent to the tensor network approximation. Two key factors govern the feasibility of scaling quantum algorithms: qubit count and gate counts, where the latter is limited by gate fidelity, at least in the NISQ era. Extrapolating the observed scaling of required QAOA layers, as shown in Fig. 8(b), a 2,700-variable JIT-JSSP instance might require a

Algorithm	Instance Size (Qubits)	Circuit Layers (p)	Total Circuit Executions	1Q Gates	2Q Gates	Backend	Best Solution	Opt. Sol. Probability
Iterative-QAOA	24	4	40,000	408	312	Ideal Sim.	**193**	0.979
	32	5	40,000	672	780	Ideal Sim.	**193**	0.780
						QPU	**193**	0.007
						QPU+DNL	**193**	0.693
	33	5	40,000	693	840	Ideal Sim.	**193**	0.914
						QPU	**193**	0.009
						QPU+DNL	**193**	0.755
	36	6	40,000	900	972	Ideal Sim. (MPS)	**193**	0.794
						QPU	198	0.000
						QPU+DNL	198	0.000
	50	6	40,000	1,250	1,686	Ideal Sim. (MPS)	198	0.000
	97	6	40,000	2,425	5,208	Ideal Sim. (MPS)	206	0.000
		7	40,000	2,813	6,076		206	0.000
LR-QAOA	24	4	4,000	408	312	Ideal Sim.	198	0.000
		25	4,000	2,424	1,950		**193**	0.004
		50	4,000	4,824	3,900		**193**	0.038
VarQITE	32	2*	1,006,200	72	46	Ideal Sim.	228	0.000
						QPU	208	0.000

Table 1: Summary of performance of Iterative-QAOA, LR-QAOA, and VarQITE across ideal noiseless simulations (statevector or MPS), raw QPU executions, and DNL error-mitigated runs. Resource requirements are quantified by circuit depth (p), gate counts (two-qubit gates are accounted as R_{ZZ} since they are native to this class of hardware), and total circuit executions, which aggregates all shots and iterations to represent the overall workload on the QPU. Performance is measured by the best (lowest-cost) solution obtained for the makespan of the JIT-JSSP problem and the sampling probability of the optimal configuration. The number of layers for VarQITE, marked with an asterisk (*), corresponds to the depth of the variational ansatz defined in Fig. 6. Optimal solutions are shown in bold.

$p \sim 10^2$ layer ansatz. Assuming quadratic scaling for the gate counts (see Fig. 8(a)), this corresponds to $\sim 10^7$ two-qubit gates, making it a promising target for fault-tolerant quantum computers. Additionally, ongoing improvements in gate fidelities, gate count pruning techniques, and the development of robust error mitigation methods [44] can accelerate progress significantly.

It is important to emphasize that Iterative-QAOA remains a heuristic algorithm and as such, the observed scaling behavior does not come with rigorous proofs. Nevertheless, exploring resource-efficient strategies, such as efficient qubit encodings [38–40] or combining classical decomposition methods [27, 75–77] that break down large problem instances into manageable sub-problems, with quantum heuristics remains a promising direction for leveraging near-term quantum devices for complex optimization problems.

5 Summary and Outlook

In this work, we developed and evaluated a quantum heuristic method, Iterative-QAOA, designed to solve the NP-hard JSSP. This algorithm combines two resource-efficient strategies: a non-variational approach using shallow, fixed-parameter QAOA circuits and an iterative warm-starting process. The core of this method is an iterative mechanism that uses thermal averages of quantum measurement outcomes from one iteration to prepare a biased, more efficient initial state for the next iteration. This mechanism progressively guides the quantum search toward the low-energy sector of the cost Hamiltonian without increasing the depth of the quantum circuit.

Through numerical simulations and hardware executions, we benchmarked Iterative-QAOA against both its LR-QAOA baseline and the Var-

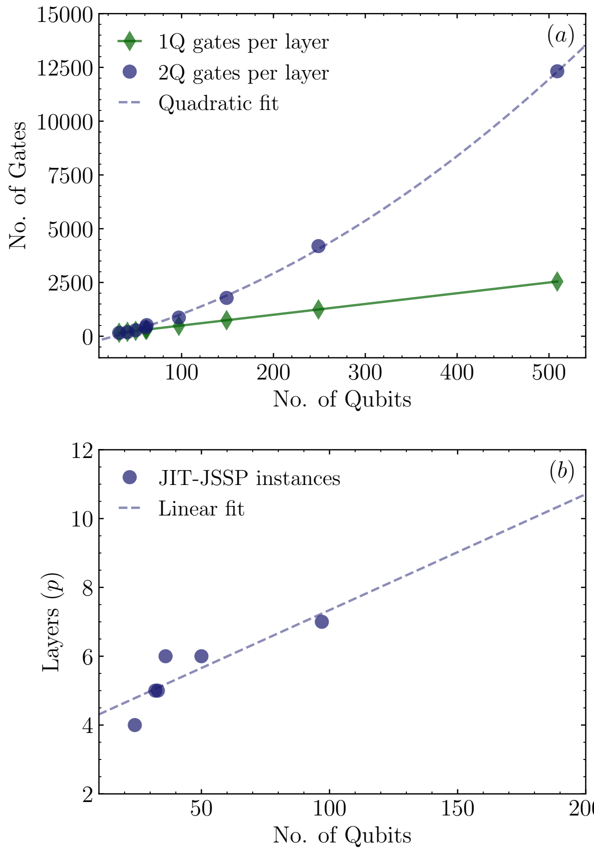

Figure 8: (a) Scaling of single- and two-qubit gate counts per QAOA layer as a function of instance size (number of qubits). Data points correspond to JIT-JSSP sub-instances derived from the 1,300-qubit full instance. The three largest points represent circuits generated solely to construct the QAOA ansatz and extract gate counts (these circuits were not simulated or executed on quantm hardware). Two-qubit gates (native R_{ZZ} on this class of hardware) follow an approximately quadratic scaling with instance size, as indicated by the dashed fit $y = 0.028x^2+10.493x-298.214$. (b) Number of QAOA layers p used for each sub-instance executed on a simulator or quantum hardware in this work, representing the circuit depth required to achieve sufficient expressivity. A linear fit (dashed line), $y = 0.034x + 3.973$, provides an approximate extrapolation of the depth expected for larger instances of comparable expressivity.

QITE algorithm. Our results confirm that a shallow-depth Iterative-QAOA significantly outperforms a deep-circuit LR-QAOA, converging to ground-state configurations with much higher probabilities. Compared to VarQITE, where we observed unfavorable scaling for this problem class, Iterative-QAOA demonstrated robust performance, finding high-quality solutions even for the 97-qubit instance. Furthermore, execution on trapped-ion quantum processors confirmed the algorithm's viability on near-term hardware where, even without significant error mitigation, it successfully identified optimal or high-quality sub-optimal solutions.

While the results presented are promising, we acknowledge that for the problem sizes studied, state-of-the-art classical heuristics can still achieve superior time-to-solution (see Appendix B). The primary value of quantum heuristics such as Iterative-QAOA, however, lies in their potential for scalability to fault-tolerant quantum computers and their ability to explore solution spaces in fundamentally different ways. Although demonstrated here on the JSSP, the algorithm provides a general framework for combinatorial optimization. A key direction for future work is to apply this iterative method to other classes of problems, particularly those for which classical solution methods are known to scale poorly.

In addition, the Iterative-QAOA framework itself offers several avenues for refinement. In this work, we used a simple linear ramp schedule with the constraint $\Delta_\beta = \Delta_\gamma$, reducing the schedule to a single hyperparameter. Exploring non-linear schedules could lead to a more faithful approximation of the underlying adiabatic evolution, potentially yielding improved performance on larger problem instances. We also used a Boltzmann distribution to compute the weights of the linear combination of solutions that constructs the initial state of the algorithm. Exploring more sophisticated methods of constructing the initial state could also potentially accelerate the convergence of the algorithm leading to a better time-to-solution. Another promising direction is the use of more efficient qubit encodings [38–40]. While such encodings could allow larger instances to be mapped on current hardware, this often comes at a cost of introducing higher-order local interactions in the Hamiltonian. Analyzing this trade-off, in the context of Iterative-QAOA, could be an interesting future research direction.

Acknowledgments

We would like to thank Felix Tripier, Yvette de Sereville, Andrew Arrasmith, and Andrii Maksymov for helpful discussions on DNL filtering and error mitigation techniques in general. This publication is supported by funding from The South Carolina Quantum Association.

References

[1] Hegen Xiong, Shuangyuan Shi, Danni Ren, and Jinjin Hu. "A survey of job shop scheduling problem: The types and models". Computers and Operations Research **142**, 105731 (2022).

[2] Alan S. Manne. "On the Job-Shop Scheduling Problem". Operations Research **8**, 219–223 (1960).

[3] A. S. Jain and S. Meeran. "Deterministic job-shop scheduling: Past, present and future". European Journal of Operational Research **113**, 390–434 (1999).

[4] Tung Kuan Liu, Yeh Peng Chen, and Jyh Horng Chou. "Solving Distributed and Flexible Job-Shop Scheduling Problems for a Real-World Fastener Manufacturer". IEEE Access **2**, 1598–1606 (2015).

[5] Jian Zhang, Guofu Ding, Yisheng Zou, Shengfeng Qin, and Jianlin Fu. "Review of job shop scheduling research and its new perspectives under Industry 4.0". Journal of Intelligent Manufacturing **30**, 1809–1830 (2019).

[6] S. Meeran and M. S. Morshed. "A hybrid genetic tabu search algorithm for solving job shop scheduling problems: A case study". Journal of Intelligent Manufacturing **23**, 1063–1078 (2012).

[7] F. Smaili. "Simulation-based study of dispatching rules in a real-world job shop scheduling problem". Journal of Simulation **00**, 1–16 (2025).

[8] Adil Baykasoğlu and Fatma S. Karaslan. "Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach". International Journal of Production Research **55**, 3308–3325 (2017).

[9] Oussama Masmoudi, Xavier Delorme, and Paolo Gianessi. "Job-shop scheduling problem with energy consideration". International Journal of Production Economics **216**, 12–22 (2019).

[10] Edilson Reis Rodrigues Kato, Gabriel Diego de Aguiar Aranha, and Roberto Hideaki Tsunaki. "A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing". Computers and Industrial Engineering **125**, 178–189 (2018).

[11] Mohammad Mahdi Ahmadian, Amir Salehipour, and T C E Cheng. "A meta-heuristic to solve the just-in-time job-shop scheduling problem". Eur. J. Oper. Res. **288**, 14–29 (2021).

[12] Stéphane Dauzère-Pérès, Junwen Ding, Liji Shen, and Karim Tamssaouet. "The flexible job shop scheduling problem: A review". European Journal of Operational Research **314**, 409–432 (2024).

[13] Imran Ali Chaudhry and Abid Ali Khan. "A research survey: Review of flexible job shop scheduling techniques". International Transactions in Operational Research **23**, 551–591 (2016).

[14] Nilsen Kundakci and Osman Kulak. "Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem". Computers and Industrial Engineering **96**, 31–51 (2016).

[15] M. R. Garey, D. S. Johnson, and Ravi Sethi. "The Complexity of Flowshop and Jobshop Scheduling". Mathematics of Operations Research **1**, 117–129 (1976).

[16] Peter Brucker, Yu N. Sotskov, and Frank Werner. "Complexity of shop-scheduling problems with fixed number of jobs: A survey". Mathematical Methods of Operations Research **65**, 461–481 (2007).

[17] Michael L. Pinedo. "Scheduling". Springer International Publishing. Cham (2022). 6th edition edition.

[18] Peter Brucker, Bernd Jurisch, and Bernd Sievers. "A branch and bound algorithm for the job-shop scheduling problem". Discrete Applied Mathematics **49**, 107–127 (1994).

[19] Wen Yang Ku and J. Christopher Beck. "Mixed integer programming models for job shop scheduling: a computational analysis". Computers and Operations Research **73**, 165–173 (2016).

[20] Bernd Waschneck, Thomas Altenmüller, Thomas Bauernhansl, and Andreas Kyek. "Production scheduling in complex job shops from an industry 4.0 perspective: A review and challenges in the semiconductor industry". In SAMI@iKNOW. (2016). url: `https://api.semanticscholar.org/CorpusID:579206`.

[21] R. J. M. Vaessens, E. H. L. Aarts, and J. K. Lenstra. "Job Shop Scheduling by Local

Search". INFORMS Journal on Computing **8**, 302–317 (1996).

[22] Wayne E. Smith. "Various optimizers for single-stage production". Naval Research Logistics Quarterly **3**, 59–66 (1956).

[23] M. A.B. Candido, S. K. Khator, and R. M. Barcia. "A genetic algorithm based procedure for more realistic job shop scheduling problems". International Journal of Production Research **36**, 3437–3457 (1998).

[24] Busra Tutumlu and Tugba Saraç. "A MIP model and a hybrid genetic algorithm for flexible job-shop scheduling problem with job-splitting". Computers and Operations Research **155**, 106222 (2023).

[25] Mauro Dell'Amico and Marco Trubian. "Applying tabu search to the job-shop scheduling problem". Annals of Operations Research **41**, 231–252 (1993).

[26] Jin Xie, Xinyu Li, Liang Gao, and Lin Gui. "A hybrid genetic tabu search algorithm for distributed flexible job shop scheduling problems". Journal of Manufacturing Systems **71**, 82–94 (2023).

[27] A I Pakhomchik, S Yudin, M R Perelshtein, A Alekseyenko, and S Yarkoni. "Solving workflow scheduling problems with QUBO modeling" (2022). arXiv:2205.04844.

[28] Costantino Carugno, Maurizio Ferrari Dacrema, and Paolo Cremonesi. "Evaluating the job shop scheduling problem on a D-wave quantum annealer". Sci. Rep. **12**, 6539 (2022).

[29] Davide Venturelli, Dominic J J Marchand, and Galo Rojo. "Quantum annealing implementation of job-shop scheduling" (2015). arXiv:1506.08479.

[30] Lukas Schmidbauer, Carlos A Riofrío, Florian Heinrich, Vanessa Junk, Ulrich Schwenk, Thomas Husslein, and Wolfgang Mauerer. "Path matters: Industrial data meet quantum optimization" (2025). arXiv:2504.16607.

[31] Berend Denkena, Fritz Schinkel, Jonathan Pirnay, and Sören Wilmsmeier. "Quantum algorithms for process parallel flexible job shop scheduling". CIRP J. Manuf. Sci. Technol. **33**, 100–114 (2021).

[32] Philipp Schworm, Xiangqian Wu, Moritz Glatt, and Jan C Aurich. "Solving flexible job shop scheduling problems in manufacturing with quantum annealing". Prod. Eng. **17**, 105–115 (2022).

[33] Philipp Schworm, Xiangqian Wu, Matthias Klar, and Jan C Aurich. "Evaluation of quantum annealing-based algorithms for flexible job shop scheduling" (2024). arXiv:2408.15671.

[34] Yize Sun, Jiarui Liu, Yunpu Ma, and Volker Tresp. "Differentiable quantum architecture search for job shop scheduling problem" (2024). arXiv:2401.01158.

[35] Krzysztof Kurowski, Tomasz Pecyna, Mateusz Slysz, Rafal Rozycki, Grzegorz Waligora, and Jan Weglarz. "Application of quantum approximate optimization algorithm to job shop scheduling problem". Eur. J. Oper. Res. **310**, 518–528 (2023).

[36] David Amaro, Matthias Rosenkranz, Nathan Fitzpatrick, Koji Hirano, and Mattia Fiorentini. "A case study of variational quantum algorithms for a job shop scheduling problem". EPJ Quantum Technol. **9**, 1–20 (2022).

[37] Archismita Dalal, Iraitz Montalban, Narendra N Hegade, Alejandro Gomez Cadavid, Enrique Solano, Abhishek Awasthi, Davide Vodola, Caitlin Jones, Horst Weiss, and Gernot Füchsel. "Digitized counterdiabatic quantum algorithms for logistics scheduling". Phys. Rev. Appl. **22**, 064068 (2024).

[38] Daniel Alexander Leidreiter. "Investigating evolving ansatz VQE algorithms for job shop scheduling". Master's thesis. Institut Für Informatik - Der Ludwig–Maximilians–Universität München. (2024). url: `https://elib.dlr.de/206087/1/Leidreiter-Investigating-Evolving-Ansatz-VQE-Algorithms-for-Job-Shop-Scheduling.pdf`.

[39] Mathias Schmid, Sarah Braun, Rudolf Sollacher, and Michael J Hartmann. "Highly efficient encoding for job-shop scheduling problems and its application on quantum computers" (2024). arXiv:2401.16381.

[40] Eric Bourreau, Gerard Fleury, and Phlippe Lacomme. "Indirect job-shop coding using rank: application to QAOA (IQAOA)" (2024). arXiv:2402.18280.

[41] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approx-

imate optimization algorithm" (2014). arXiv:1411.4028.

[42] Edward Farhi and Aram W Harrow. "Quantum supremacy through the quantum approximate optimization algorithm" (2016). arXiv:1602.07674.

[43] Ashley Montanaro and Leo Zhou. "Quantum speedups in solving near-symmetric optimization problems by low-depth QAOA" (2024). arXiv:2411.04979.

[44] Ruslan Shaydulin, Changhao Li, Shouvanik Chakrabarti, Matthew DeCross, Dylan Herman, Niraj Kumar, Jeffrey Larson, Danylo Lykov, Pierre Minssen, Yue Sun, Yuri Alexeev, Joan M Dreiling, John P Gaebler, Thomas M Gatterman, Justin A Gerber, Kevin Gilmore, Dan Gresh, Nathan Hewitt, Chandler V Horst, Shaohan Hu, Jacob Johansen, Mitchell Matheny, Tanner Mengle, Michael Mills, Steven A Moses, Brian Neyenhuis, Peter Siegfried, Romina Yalovetzky, and Marco Pistoia. "Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem". Sci. Adv. **10**, eadm6761 (2024).

[45] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. "Barren plateaus in quantum neural network training landscapes". Nat. Commun. **9**, 4812 (2018).

[46] Daniel J Egger, Jakub Mareček, and Stefan Woerner. "Warm-starting quantum optimization". Quantum **5**, 479 (2021). arXiv:2009.10095v4.

[47] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti. "An initialization strategy for addressing barren plateaus in parametrized quantum circuits" (2019). arXiv:1903.05076.

[48] Dennis Willsch, Madita Willsch, Fengping Jin, Kristel Michielsen, and Hans De Raedt. "GPU-accelerated simulations of quantum annealing and the quantum approximate optimization algorithm". Comput. Phys. Commun. **278**, 108411 (2022).

[49] Stefan H Sack and Maksym Serbyn. "Quantum annealing initialization of the quantum approximate optimization algorithm". Quantum **5**, 491 (2021). arXiv:2101.05742v3.

[50] Yunlong Yu, Chenfeng Cao, Carter Dewey, Xiang-Bin Wang, Nic Shannon, and Robert Joynt. "Quantum approximate optimization algorithm with adaptive bias fields". Phys. Rev. Res. **4**, 023249 (2022).

[51] Yunlong Yu, Chenfeng Cao, Xiang-Bin Wang, Nic Shannon, and Robert Joynt. "Solution of SAT problems with the adaptive-bias quantum approximate optimization algorithm". Phys. Rev. Res. **5**, 023147 (2023).

[52] Linghua Zhu, Ho Lun Tang, George S Barron, F A Calderon-Vargas, Nicholas J Mayhall, Edwin Barnes, and Sophia E Economou. "Adaptive quantum approximate optimization algorithm for solving combinatorial problems on a quantum computer". Phys. Rev. Res. **4**, 033029 (2022).

[53] Reuben Tate, Jai Moondra, Bryan Gard, Greg Mohler, and Swati Gupta. "Warm-started QAOA with custom mixers provably converges and computationally beats goemans-williamson's max-cut at low circuit depths". Quantum **7**, 1121 (2023). arXiv:2112.11354v4.

[54] Rafael S do Carmo, Marcos C S Santana, Felipe F Fanchini, Victor Hugo C de Albuquerque, and João Paulo Papa. "Warm-starting QAOA with XY mixers: A novel approach for quantum-enhanced vehicle routing optimization" (2025). arXiv:2504.19934.

[55] Yunlong Yu, Xiang-Bin Wang, Nic Shannon, and Robert Joynt. "Warm-start adaptive-bias quantum approximate optimization algorithm". Phys. Rev. A **112**, 012422 (2025).

[56] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices". Phys. Rev. X. **10**, 021067 (2020).

[57] Fernando G S L Brandao, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. "For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances" (2018). arXiv:1812.04170.

[58] Maximilian Hess, Lilly Palackal, Abhishek Awasthi, and Karen Wintersperger. "Effective embedding of integer linear inequalities for variational quantum algorithms" (2024). arXiv:2403.18395.

[59] Ryo Sakai, Hiromichi Matsuyama, Wai-Hong Tam, and Yu Yamashiro. "Transferring linearly fixed QAOA angles: performance and real device results" (2025). arXiv:2504.12632.

[60] Ruslan Shaydulin, Stuart Hadfield, Tad Hogg, and Ilya Safro. "Classical symmetries and the quantum approximate optimization algorithm". Quantum Inf. Process. **20**, 1–28 (2021).

[61] Vladimir Kremenetski, Anuj Apte, Tad Hogg, Stuart Hadfield, and Norm M Tubman. "Quantum alternating operator ansatz (QAOA) beyond low depth with gradually changing unitaries" (2023). arXiv:2305.04455.

[62] J A Montañez-Barrera and Kristel Michielsen. "Toward a linear-ramp QAOA protocol: evidence of a scaling advantage in solving some combinatorial optimization problems". Npj Quantum Inf. **11**, 1–12 (2025).

[63] Vanessa Dehn, Martin Zaefferer, Gerhard Hellstern, Florentin Reiter, and Thomas Wellens. "Extrapolation method to optimize linear-ramp QAOA parameters: Evaluation of QAOA runtime scaling" (2025). arXiv:2504.08577.

[64] Ruslan Shaydulin and Marco Pistoia. "QAOA with $n \cdot p \geq 200$" (2023). arXiv:2303.02064.

[65] Jonathan Wurtz and Danylo Lykov. "Fixed-angle conjectures for the quantum approximate optimization algorithm on regular MaxCut graphs". Phys. Rev. A **104**, 052419 (2021).

[66] Haomu Yuan, Songqinghao Yang, and Crispin H W Barnes. "Iterative quantum optimisation with a warm-started quantum state" (2025). arXiv:2502.09704.

[67] Alejandro Gomez Cadavid, Archismita Dalal, Anton Simen, Enrique Solano, and Narendra N Hegade. "Bias-field digitized counterdiabatic quantum optimization". Phys. Rev. Res. **7**, L022010 (2025).

[68] Xiao Yuan, Suguru Endo, Qi Zhao, Ying Li, and Simon C Benjamin. "Theory of variational quantum simulation". Quantum **3**, 191 (2019). arXiv:1812.08767v4.

[69] Sam McArdle, Tyson Jones, Suguru Endo, Ying Li, Simon C Benjamin, and Xiao Yuan. "Variational ansatz-based quantum simulation of imaginary time evolution". Npj Quantum Inf. **5**, 1–6 (2019).

[70] Titus D Morris, Ananth Kaushik, Martin Roetteler, and Phillip C Lotshaw. "Performant near-term quantum combinatorial optimization" (2024). arXiv:2404.16135.

[71] K Mitarai, M Negoro, M Kitagawa, and K Fujii. "Quantum circuit learning". Phys. Rev. A **98**, 032309 (2018).

[72] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. "Evaluating analytic gradients on quantum hardware". Phys. Rev. A **99**, 032331 (2019).

[73] Guifré Vidal. "Efficient classical simulation of slightly entangled quantum computations". Phys. Rev. Lett. **91**, 147902 (2003).

[74] Ulrich Schollwöck. "The density-matrix renormalization group in the age of matrix product states". Ann. Phys. **326**, 96–192 (2011).

[75] Moises Ponce, Rebekah Herrman, Phillip C Lotshaw, Sarah Powers, George Siopsis, Travis Humble, and James Ostrowski. "Graph decomposition techniques for solving combinatorial optimization problems with variational quantum algorithms" (2023). arXiv:2306.00494.

[76] Mohammed M S El-Kholany, Martin Gebser, and Konstantin Schekotihin. "Decomposition strategies and multi-shot ASP solving for job-shop scheduling" (2022). arXiv:2205.07537.

[77] S David Wu, Eui-Seok Byeon, and Robert H Storer. "A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness". Oper. Res. **47**, 113–124 (1999).

[78] Jwo-Sy Chen, Erik Nielsen, Matthew Ebert, Volkan Inlek, Kenneth Wright, Vandiver Chaplin, Andrii Maksymov, Eduardo Páez, Amrit Poudel, Peter Maunz, et al. "Benchmarking a trapped-ion quantum computer with 30 qubits". Quantum **8**, 1516 (2024).

[79] Sangtaek Kim, Robert R. McLeod, M. Saffman, and Kelvin H. Wagner. "Doppler-free, multiwavelength acousto-optic deflector for two-photon addressing arrays of Rb atoms in a quantum information processor". Appl. Opt. **47**, 1816–1831 (2008).

[80] I. Pogorelov, T. Feldker, Ch. D. Marciniak, L. Postler, G. Jacob, O. Krieglsteiner, V. Podlesnic, M. Meth, V. Negnevitsky, M. Stadler, et al. "Compact Ion-Trap Quantum Computing Demonstrator". PRX Quantum **2**, 020343 (2021).

[81] Ivan A Chernyshev, Roland C Farrell, Marc Illa, Martin J Savage, Andrii Maksymov, Felix Tripier, Miguel Angel Lopez-Ruiz, Andrew Arrasmith, Yvette de Sereville, Aharon Brodutch, Claudio Girotto, Ananth Kaushik, and Martin Roetteler. "Pathfinding quantum simulations of neutrinoless double-β decay" (2025). arXiv:2506.05757.

A JSSP Instance Generation

Parameter	Symbol	Value
Machines	M	3
Jobs	J	20
Time Slots	T_m	{20, 22, 23}
Earliness Cost	c_e	1
Lateness Cost	c_l	3
Production Switching Cost	c_p	5
Penalty Weight	λ	10

Table 2: Parameters for the full JIT-JSSP instance used in the experimental validation. The meaning of the variables are explained in Section 4.1.

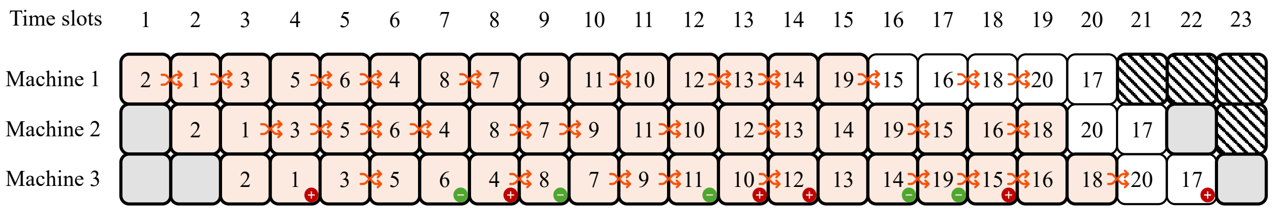

Figure 9: Gantt chart representing an optimal solution to the full instance and the time slots unfrozen in the 33-qubit instance. Each row corresponds to a machine and each column is a time slot that has an assigned job. Orange boxes represent fixed job assignments, gray boxes represent the fixed empty (idle) time slots assignments, while white boxes are free (unfrozen) time periods that are optimized. Dashed boxes denote empty time slots which were not classically optimized. Unit penalties are accrued for each production group switch (crossing arrows), early deliveries (green minus signs), and late deliveries (red plus signs). The production groups and due times of the underlying JIT-JSSP instance are defined in Table 3.

	Machine	Jobs																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Production Groups	1	A	B	C	D	C	B	C	D	C	B	C	D	A	B	C	D	D	B	C	D
	2	A	A	B	B	C	D	A	B	D	C	D	C	D	D	A	A	B	B	D	B
	3	B	B	B	A	A	A	C	C	B	D	D	A	A	A	B	D	B	D	C	B
Due Times	–	3	3	5	6	6	8	10	11	11	12	13	13	15	17	17	19	20	20	21	21

Table 3: Production groups and due times of the full JIT-JSSP instance.

We designed a problem instance with $J = 20$ jobs, each of which had to be processed on $M = 3$ machines. On machine 1, all time slots were active (i.e., processed a job), and there were no idle slots. Following this, the number of time slots needed for machine 1 was $T_1 = J = 20$. On machine 2, two

Free Variables (n_{var})	Machine 1			Machine 2			Machine 3		
	Jobs	Time Slots	n_1	Jobs	Time Slots	n_2	Jobs	Time Slots	n_3
24	16-18, 20	17-20	4	17, 20	20-21	2	17, 20	21-22	2
32*	15-18, 20	16-20	5	17, 20	20-21	2	17, 20	21-22	2
33	15-18, 20	16-20	5	17, 20	20-21	2	17, 20	21-22	2
36	16-18, 20	17-20	4	16-18, 20	18-21	4	17, 20	21-22	2
50	15-18, 20	16-20	5	16-18, 20	18-21	4	17-18, 20	20-22	3
97	15-20	15-20	6	15-20	16-21	6	15-18, 20	18-22	5

Table 4: Construction of JIT-JSSP sub-instances. Free (unfrozen) jobs and time slots used to generate all instances evaluated in this work. The number of free jobs/time slots at the end of the schedule for each machine (n_1, n_2, n_3) follows Eq. (24), with the condition that $n_1 \geq n_2 \geq n_3$. The 32-variable instance (*) is derived from the 33-variable instance by additionally fixing variable $x_{1,15,20}$ to its optimal value of 0. The Gantt chart representing the 33-variable instance is shown in Fig. 9.

time slots were allowed to be idle, corresponding to $T_2 = J + 2 = 22$, and on machine 3, three time slots were allowed to be idle, corresponding to $T_3 = J + 3 = 23$.

The original formulation we solved is similar to the formulation presented in Section 4.1, with one key difference: it did not exogenously specify which time slots on each machine were idle. That is, it did not include sets of active (A_m) and idle (I_m) time slots for each machine m; rather, it followed the time assignment and idle slot constraints as presented in Ref. [36]. Solving the original instance produced the optimal solution presented in Fig. 9. After the original instance was generated, we used it to exogenously define the sets of active and idle time slots on each machine, essentially pre-defining whether a time slot on a machine would be active or idle. The resulting sets of idle time slots were $I_1 = \emptyset, I_2 = \{1, 22\}, I_3 = \{1, 2, 23\}$ (the grey boxes on Fig. 9) and active time slots on each machine m were given by $A_m = \{1, \ldots, T_m\} \setminus I_m$ (the orange and white boxes on Fig. 9). We used the resulting formulation with exogenously defined sets of active and idle slots (given in Section 4.1) for all analyses and discussion, and we refer to it throughout the paper as the "full instance." Note that the optimal job-machine assignments in the original and full instances are the same.

We solved the full instance classically using Gurobi Optimizer software version 12.0.2 on an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz. This problem had 1,300 binary variables and 23 continuous variables; the continuous variables represent the due date and production cost penalties, far too many to be solved via a quantum simulation or current quantum hardware. In order to end up with problems of suitable sizes for our simulator and hardware, we designed several sub-instances, where we strategically "froze" most variables in the full instance to their optimal value, but left a subset of the variables unfrozen, so that we could solve each sub-instance for these variables using quantum simulation and/or hardware. The free slots remain fixed once the full instance is solved, so each sub-instance only considers a subset of the x_{mjt} variables. Ref. [36] followed a similar process of freezing most variables, then solving for the unfrozen variables using quantum technology. Our goal was to generate a set of sub-instances that covered a wide range of number of variables, so that we could assess the performance of Iterative-QAOA across a range of problem sizes. The process of how we designed each sub-instance and determined the associated number of unfrozen variables is detailed below.

1. We chose the number of slots (and therefore, the same number of jobs) to leave unfrozen on each machine, abiding by the following constraints:
   (a) We always freed up jobs at the end of each machine's time slots.
   (b) The number of free slots on machine 1 $\geq$ the number of free slots on machine 2 $\geq$ the number of free slots on machine 3.

   Note: These rules served two purposes. First, they prevented us from choosing trivial cases where visual inspection of the problem would allow it to be reduced to a smaller number of variables.

Second, they ensured that we could use the same formula (shown in step 2) to calculate the number of variables in each sub-instance.

2. We calculated the number of unfrozen variables associated with this set of free slots as follows

$$n_1^2 + n_2^2 + n_3^2 = n_{\text{var}}. \tag{24}$$

This result is fairly intuitive, on machine i, we have n_i free jobs, each of which can be placed in any of n_i slots; therefore, we need to have an unfrozen variable for each potential job/slot pair, so we have n_1^2 variables for machine 1, n_2^2 variables for machine 2, and n_3^2 variables for machine 3.

3. If the problem size calculated in step 2 was useful for us, we added it to the list of instances and returned to the first step to generate other instances as needed. If it was not useful for us, we returned to the first step to generate other instances. When our goal was to arrive at a specific number of variables, and this number couldn't be achieved exactly via steps 1 and 2, we proceeded to step 4.

4. Starting with the smallest sub-instance with more variables than our desired number, we refroze variables until we arrived at our desired number of variables (note that n_{var} decreased by 1 for each variable that we refroze). We refroze variables whose optimal value was 0, because freezing variables whose optimal value is 1 would have resulted in a much less interesting problem. We now have our set of unfrozen variables, and this sub-instance is complete.

In Table 2, we show all the parameter values used to construct the cost function (Eq. (22)) of the full problem instance, and Table 3 shows the production groups and due dates.

To illustrate the instance construction process, Fig. 9 displays a specific sub-problem overlaid on an optimal solution of the full instance. In the Gantt chart, orange boxes represent variables fixed (frozen) to the their optimal values, while white boxes, denote the free (unfrozen) variables that define the sub-instance to be solved. Dashed boxes indicate empty time slots, arrows mark production group switches, and the plus and minus signs represent a late/early delivery of the job on machine 3, respectively. In this example, the number of free slots (white boxes) per machine are $n_1 = 5$, $n_2 = 2$, and $n_3 = 2$. According to Eq. (24), this configuration corresponds to $n_{\text{var}} = 33$ free binary variables. The 33-variable cost function (Eq. (22)) is then mapped onto a 33-qubit Hamiltonian (one-hot encoding) using the transformation Eq. (5). In Table 4 we show the specific configurations of unfrozen jobs and time slots utilized to generate all sub-instances used in this work.

B Classical Solvers

The JIT-JSSP instances we consider in this paper can also be solved using classical hardware. We present the time to solve to optimality using the Gurobi Optimizer (with presolve) software version 12.0.2 solved on a PC with Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz. Appendix B shows the solution times to solve sub-instances with progressively fewer frozen variables. The time horizon $[t, T_m]$ associated with each row represents the time slots on machine m that are optimized in a given sub-instance, i.e., with unfrozen variables; all other variables are frozen to the optimal solution presented in Fig. 9. Note that the row with the time horizon of $[1, T_m]$ corresponds to the full instance. Exponential scaling in computational time as a function of instance size is shown in Fig. 10.

C LR-QAOA Parameter Exploration

The performance of LR-QAOA is primarily determined by the ramp parameters Δ_β and Δ_γ and the number of layers p. To simplify the parameter search, we assume equal slopes, defining $\Delta_{\beta/\gamma} \equiv \Delta_\beta = \Delta_\gamma$. The optimal value of $\Delta_{\beta/\gamma}$ is identified by computing the expectation value of the cost Hamiltonian $\langle H_C \rangle$, from 4,000 measurement shots in ideal simulations for each point in the parameter landscape.

Time horizon $[t, T_m]$	Free Variables (n_{var})	Computational time [sec]			
		Run 1	Run 2	Run 3	Avg.
$[17, T_m]$	88	0.03	0.03	0.03	0.03
$[16, T_m]$	123	0.04	0.05	0.03	0.04
$[15, T_m]$	164	0.08	0.07	0.07	0.07
$[14, T_m]$	211	0.10	0.08	0.11	0.10
$[13, T_m]$	264	0.21	0.12	0.11	0.15
$[12, T_m]$	323	0.33	0.29	0.53	0.39
$[11, T_m]$	388	0.51	0.48	0.46	0.48
$[10, T_m]$	459	1.30	1.09	1.10	1.16
$[9, T_m]$	536	2.14	2.03	1.96	2.04
$[8, T_m]$	619	2.28	2.36	2.46	2.36
$[7, T_m]$	708	3.73	3.59	3.35	3.56
$[6, T_m]$	803	7.95	9.18	9.34	8.82
$[5, T_m]$	904	28.25	32.83	32.56	31.21
$[4, T_m]$	1011	53.28	41.31	33.37	42.65
$[3, T_m]$	1,124	112.62	98.76	131.12	114.17
$[2, T_m]$	1,221	124.64	129.96	103.53	119.38
$[1, T_m]$	1,300	226.00	229.74	229.93	228.56

Table 5: Time to solve to optimality on a classical computer for sub-instances of the full instance. The unfrozen (free) binary variables in each sub-instance are the binary decision variables between t and T_m (inclusive) that are optimized; this excludes any decision variable associated with a job j assigned to machine m in a time period prior to t.

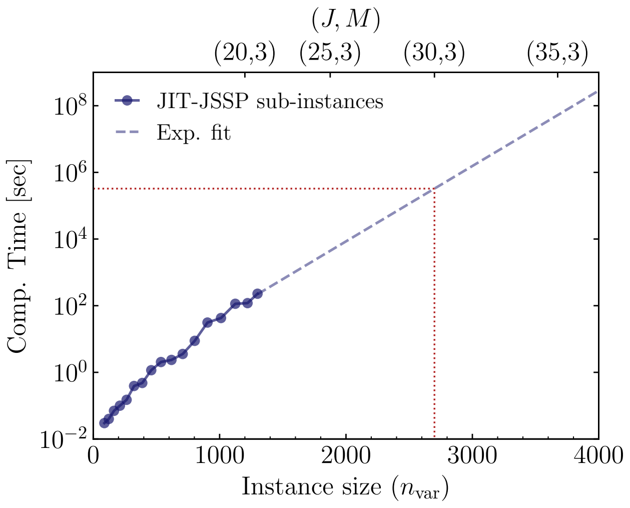

Figure 10: Classical computational time for sub-instances beginning at time period t. The data points correspond to the mean time-to-solution values from Appendix B. The dashed line shows an exponential fit, extrapolating the expected computational time for larger problems. The extrapolated computational time is only an approximate guide and will need to be verified by actual execution. The upper horizontal axis shows example combinations of jobs J and machines M that correspond to the lower bound of instance size J^2M. The dotted red line indicates a problem size for which the current formulation of the JSSP would take around 4 days to execute. This could represent a point beyond which quantum computation may begin to have advantage over classical computation.

In Fig. 11 we present the landscapes for three JIT-JSSP instances of increasing size: 12, 24, and 32 qubits. For visual clarity, the color scale in each panel is normalized as

$$\frac{\langle H_C \rangle}{\max \langle H_C \rangle}, \tag{25}$$

where $\max\langle H_C\rangle$ denotes the maximum expectation value computed across all values of $\Delta_{\beta/\gamma}$ and p for a given problem instance; ensuring a uniform z-scale across all panels.

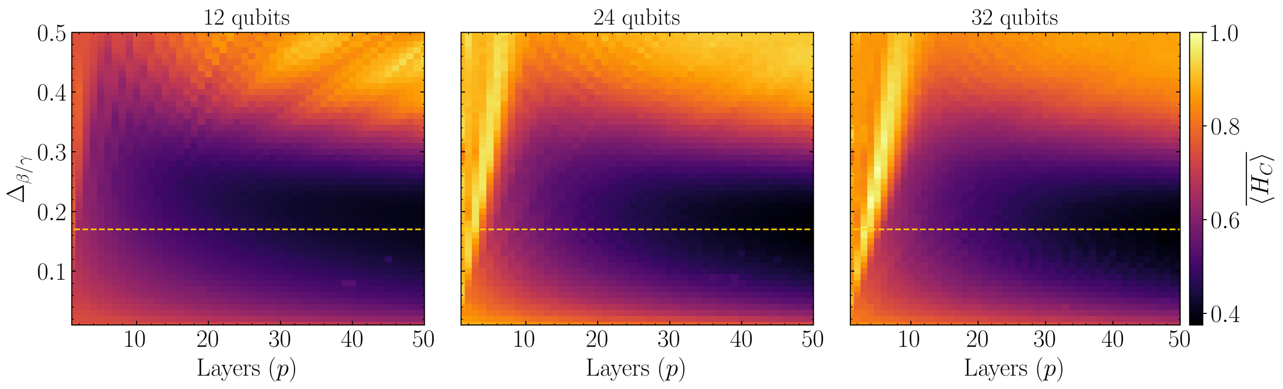

Figure 11: LR-QAOA performance landscape for JIT-JSSP instances of increasing size. Each panel correspond to a different problem instance: 12, 24, and 32 qubits (from left to right). The color scale shows the average cost $\langle H_C \rangle$ normalized by $\max\langle H_C \rangle$ as a function of QAOA layers p and ramp parameter $\Delta_{\beta/\gamma}$. The dashed horizontal line at $\Delta_{\beta/\gamma} = 0.17$ highlights the optimal parameter trajectory for all three instances.

A key feature across all three cases is the emergence of a clear "performance valley" [61, 62], a region where the algorithm most effectively minimizes the cost Hamiltonian as circuit depth increases. Notably, the optimal trajectory along this valley is consistently located at a ramp parameter of $\Delta_{\beta/\gamma} \approx 0.17$, regardless of the problem size. We selected this value for all experiments in the main text, assuming this optimal parameter holds or deviates only slightly for larger instances.

While the normalization $\langle H_C \rangle / \max\langle H_C \rangle$ helps visualize the "performance valley", the absolute performance of LR-QAOA is still size-dependent. The lowest average energy $\min\langle H_C \rangle$, increases as system size grows for a fixed p. This shows how the complexity of the problem increases with system size, therefore requiring deeper circuits to reach high-quality solutions.

D Performance with sub-optimal LR-QAOA parameters

To study robustness, we evaluated the performance of Iterative-QAOA using a sub-optimal parameter schedule, selecting a ramp parameter of $\Delta_{\beta/\gamma} = 1.25$; a value located far from the optimal performance valley (see Fig. 11).

The results of these simulations and hardware runs for the same 32-, 33-, and 36-qubit instances are shown in Figs. 12 and 13. Like in the main text, the two smallest instances were simulated using a statevector simulator, while the 36-qubit instance ideal results were obtained using an MPS simulator with a maximum bond dimension of $\chi = 256$. Despite the sub-optimal schedule, the iterative process is still able to effectively drive the cost distributions towards the low-energy sector, successfully sampling the optimal solution in both ideal noiseless simulations and quantum hardware runs. While the final probabilities of sampling the ground state are lower than those achieved with the optimal schedule, this result underscores the robustness of the algorithm. It demonstrates that the iterative refinement of the initial state is a powerful mechanism that can compensate for a poorly chosen parameter schedule.

E Trapped-ion Quantum Hardware

The experiments were performed on IonQ's Forte-generation trapped-ion quantum processors, specifically the Forte and Forte Enterprise systems [78]. Both devices employ chains of 36 $^{171}\text{Yb}^+$ ions, with quantum information encoded in the ground state hyperfine levels. Ions are produced via laser ablation and photoionization, and confined in a surface linear Paul trap integrated within a vacuum package.

Universal control is achieved through two-photon Raman transitions driven by 355 nm pulsed lasers, enabling arbitrary single-qubit rotations and native R_{ZZ}-type entangling gates. At the time of experi-

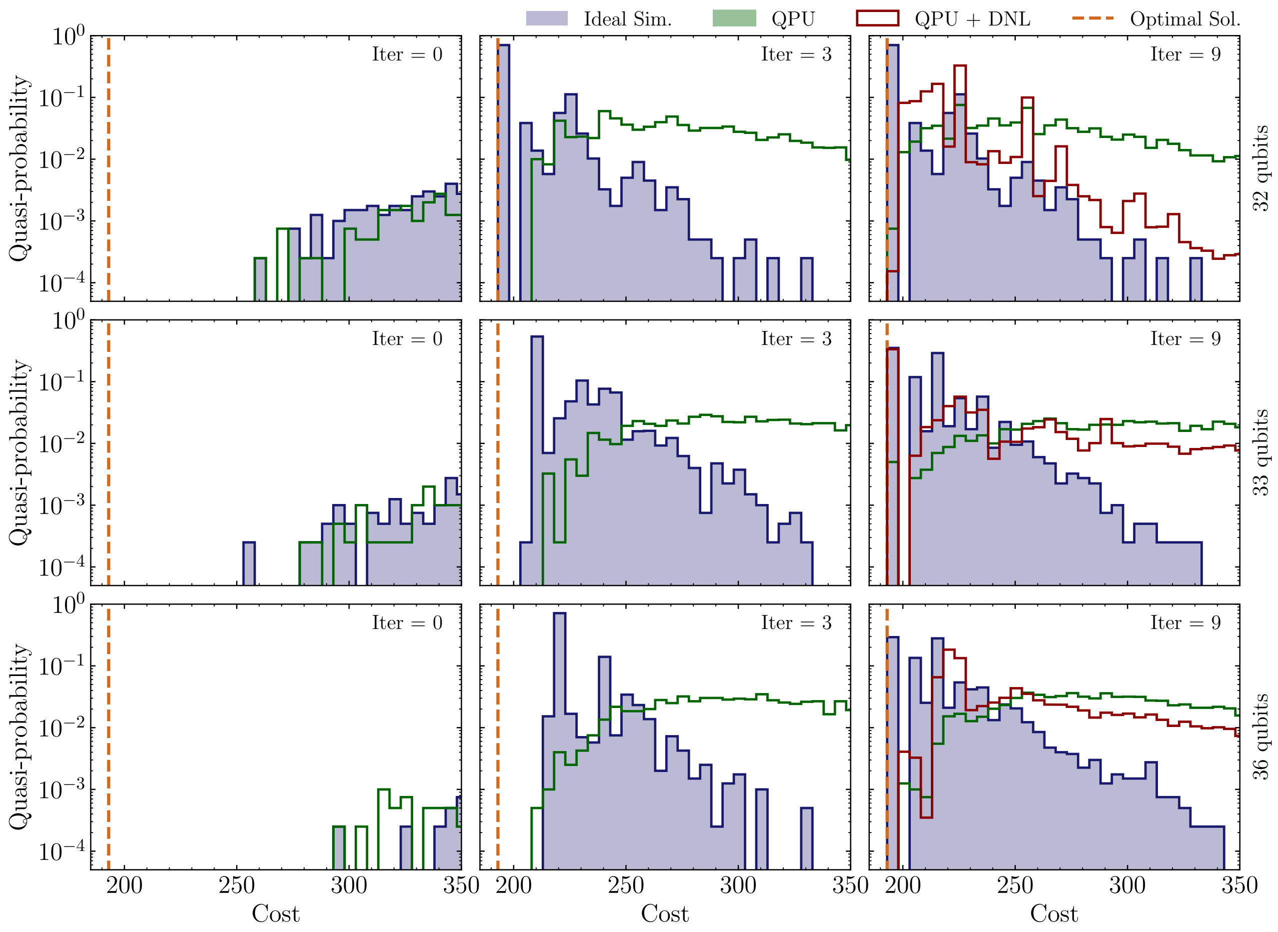

Figure 12: Performance of Iterative-QAOA on IonQ Forte Generation QPU on problems with 32, 33 and 36 qubits. These problems were executed with a sub-optimal LR-QAOA ramp parameter of $\Delta_{\beta/\gamma} = 1.25$ and a fixed $\beta_T = 0.5$. Even with sub-optimal parameters, the algorithm is robust enough to sample the optimal solution for the 32 qubits and 33 qubits problem instances and the first excited state in the 36 qubit problem instance when executed on quantum hardware.

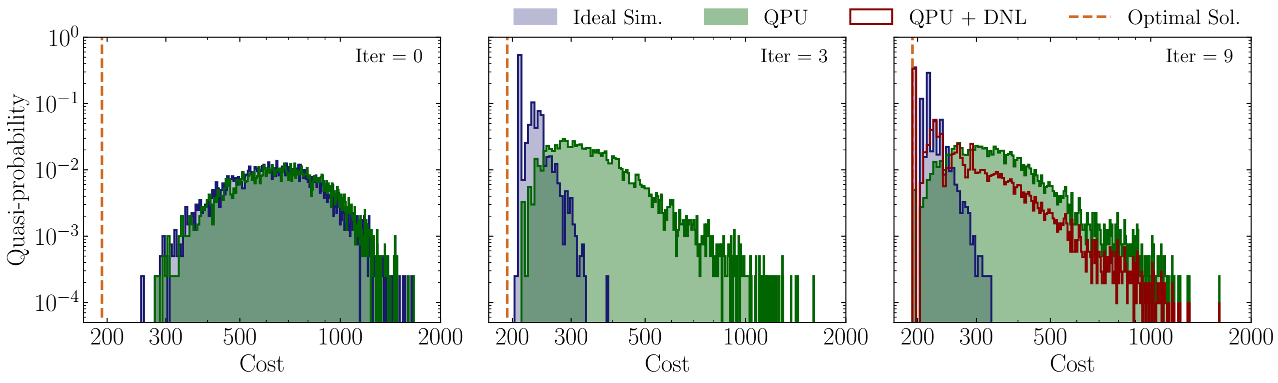

Figure 13: Performance of Iterative-QAOA on a 33-qubit JIT-JSSP instance executed on IonQ Forte generation QPU with a sub-optimal LR-QAOA ramp parameter of $\Delta_{\beta/\gamma} = 1.25$ and a fixed $\beta_T = 0.5$.

mentation, direct randomized benchmarking (DRB) reported median two-qubit gate fidelities of 99.3% (Forte) and 99.17% (Forte Enterprise), with single-qubit fidelities near 99.98%. Gate durations were approximately 130 µs for single-qubit and 950 µs for two-qubit operations.

A key architectural feature is the use of acousto-optic deflectors (AODs) for independent beam steering, which reduces crosstalk and alignment errors [79, 80]. Combined with automated calibration and control protocols, this enables robust high-fidelity operation across extended ion chains.

F Error Mitigation: Debiasing with Non-Linear Filtering

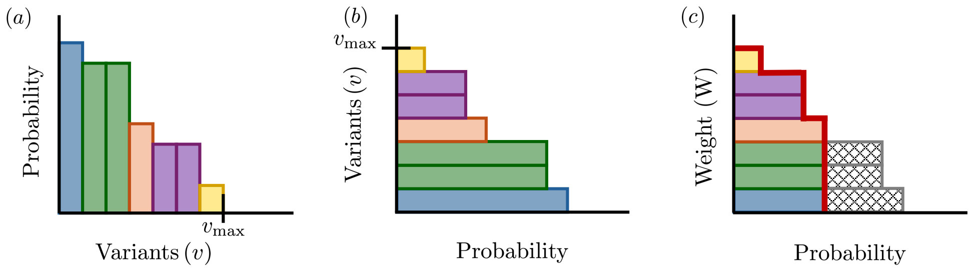

Figure 14: Aggregation process with non-linear filtering for a single measured bitstring [81]. (a) The distribution of the bitstring's probabilities is collected from each of the symmetrized circuit variants. (b) These probabilities are then sorted and converted into a survival plot, which shows the number of variants that observed the bitstring at or above a given probability. The normalized area under this curve corresponds to the unmitigated probability, which is equivalent to a simple average of all variant outcomes. (c) Finally, a non-linear filter is applied by setting a threshold v_{th}, discarding bitstrings observed by fewer than a minimum number of variants. This step removes likely noise-induced artifacts that are inconsistent across the different circuit realizations, and the final, mitigated probability is calculated from the area under the filtered curve.

In this appendix we summarize the Debiasing with Non-linear Filtering (DNL) method used for error mitigation. DNL is a resource-efficient technique that suppresses hardware-induced errors by exploiting circuit and device symmetries. The method relies on executing multiple "variants" of a target circuit, which are equivalent in the noiseless limit but differ in their physical implementation. Variants can be generated, for example, by remapping logical qubits to different physical ions, using different gate decompositions, or by pairing circuits that differ only by final NOT gates. This construction symmetrizes dominant error sources such as location-dependent noise or measurement bias, enabling their suppression in post-processing.

The DNL aggregation algorithm goes as follows. For each measured bitstring, record and order its probability distribution across all variants (Fig. 14a). Obtain the number of variants as a function of the measured probabilities of the bitstring and normalize the area under the curve (Fig. 14b). Apply a non-linear weighting filter $W(v)$ (Fig. 14c) which discards any bitstring observed by fewer than a minimum number of variants v_{th}. The specific weighting function used in this work is given by

$$W(v) = \begin{cases} \left(\dfrac{v}{v_{\max}}\right)^{\alpha}, & v > v_{\text{th}}, \\ 0, & v \leq v_{\text{th}}, \end{cases} \tag{26}$$

where $v_{\max}$ is the total number of variants and $\alpha > 0$ controls how sharply outcomes observed in fewer variants are attenuated; we use $\alpha = 4$. The final, mitigated probability for each bitstring is then calculated as a weighted average of its frequencies, with the weight for each bitstring determined by $W(v)$. This approach preserves bitstrings that are consistently observed across many variants while suppressing those that appear in only a few, which are treated as likely noise artifacts.