Commute Hamiltonian-Based QAOA for Binary Optimization under Linear Constraints

Liqiang Lu and Debin Xiang* and Qifan Jiang and Siwei Tan and Jianwei Yin*

***Abstract*—Constrained binary optimization aims to find an optimal assignment to minimize or maximize the objective meanwhile satisfying the constraints, which is a representative NP problem in various domains, including transportation, scheduling, and economics. Quantum approximate optimization algorithms (QAOA) provide a promising methodology for solving this problem by exploiting the parallelism of quantum entanglement. However, existing QAOA approaches based on penalty-term or Hamiltonian simulation fail to thoroughly encode the constraints, leading to extremely low success rates and long searching latency.**

This paper proposes Choco-Q, a formal and universal framework for constrained binary optimization problems, which comprehensively covers all constraints and exhibits high deployability for current quantum devices. The main innovation of Choco-Q is to embed the commute Hamiltonian as the driver Hamiltonian, resulting in a much more general encoding formulation that can deal with arbitrary linear constraints. Leveraging the arithmetic features of the commute Hamiltonian, we propose three optimization techniques to squeeze the overall circuit complexity, including Hamiltonian serialization, equivalent decomposition, and mid-measurement and post-selection. The serialization mechanism transforms the original Hamiltonian into smaller ones. Our decomposition methods only take linear time complexity, achieving end-to-end acceleration. Experiments demonstrate that Choco-Q shows more than 157× algorithmic improvement in successfully finding the optimal solution and achieves 4.69× end-to-end acceleration, compared to prior QAOA designs.

***Index Terms*—Quantum Computing, Constrained Binary Optimization, Variational Quantum Algorithm**

I. Introduction

The constrained binary optimization is to find an assignment of binary variables that minimizes or maximizes the objective function under the constraints. Typically, the constraints usually comprise multiple linear equations, namely equality, e.g., $C\vec{u} = \vec{c}$. There are various representative applications of this problem, such as facility location [1], project scheduling [2], portfolio optimization [3], political districting [4], and energy system optimization [5]. The constrained binary optimization has been demonstrated as an NP-hard problem that takes exponential time and space complexity for the classical solver to get the exact solution [6].

* Corresponding authors.

Liqiang Lu, Debin Xiang, Qifan Jiang, Siwei Tan, and Jianwei Yin are with the College of Computer Science, Zhejiang University, Hangzhou, China, 310058. (e-mail:{liqianglu, db.xiang, jiang7f, siweitan}@zju.edu.cn, zjuyjw@cs.zju.edu.cn)

Manuscript received xxx xx, 20xx; revised xxx xx, 20xx. This work was supported by the National Natural Science Foundation of China (No.62472374) and the National Key Research and Development Program of China (No. 2023YFF0905200). This work was also funded by the Zhejiang Pioneer (Jianbing) Project (No. 2023C01036).

TABLE I
QAOA DESIGNS FOR CONSTRAINED BINARY OPTIMIZATION.

Constraint encoding	**Penalty-term-based**		**Driver-Hamiltonian-based**	
Methods	A. Verma et al. [7]	Red-QAOA [8]	Yoshioka et al. [9] *cyclic Hamilt.*	Choco-Q *commute Hamilt.*
Universality	soft const.	soft const.	hard const. only part of linear	hard const. arbitrary linear
In-constraints rate	0.03%	0.07%	0.67%	100%
Success rate	0.02%	0.03%	0.14%	67.1%
Latency	16.6s	16.7s	19.6s	7.07s

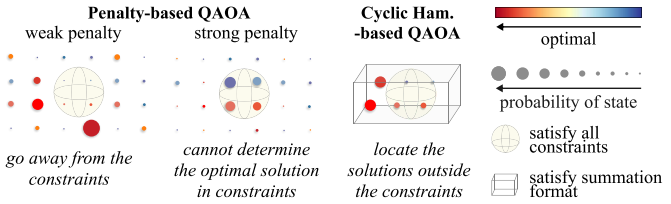
* The latency includes computing time and compilation time of QAOA, w/o data communication time. The execution time is estimated based on the IBM Fez model [10].

Quantum computing stands as a revolutionary paradigm to handle this problem by taking advantage of quantum entanglement and superposition. Notably, the quantum approximate optimization algorithm (QAOA) is a class of variational algorithms, which consists of objective Hamiltonian and driver Hamiltonian to iteratively search the optimal solution [7], [8], [11], [12]. Mathematically, it encodes the objective function and constraints into Hamiltonians and updates the parameters during simulation. In particular, the penalty-based method is a natural idea that inserts the constraint into the objective function as penalty terms [7], [13]. While, another approach is to encode the constraints into the driver Hamiltonian [9], [14].

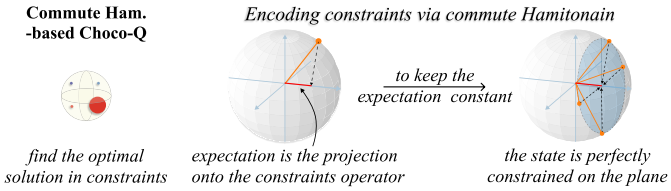
However, existing algorithms are severely limited by the low success rate, originating from the inability to thoroughly encode the constraints. Table I summarizes several features of previous QAOA designs and compares them on the graph coloring problem [15] using a 15-qubit simulator. Here, we focus on two critical metrics: 1) *in-constraints rate*, the probability that the output solutions satisfy all constraints; 2) *success rate*, the probability of getting the optimal solution. We observe an extremely low success rate across all prior methods. Especially, the penalty-based methods even exhibit a close-to-zero success rate, smashing quantum advantages.

Fundamentally, the penalty-based method only supports soft constraint encoding, which highly depends on the coefficient of the penalty terms. Concretely, a small penalty coefficient may fail to take the constraints into account, as shown in Fig. 1 (a). Nevertheless, a high coefficient will diminish the gap between the optimal solution and non-optimal ones when calculating the objective, decreasing the success rate. Thus, tuning the penalty coefficient gives rise to lots of time involved in classical computing. For example, in Table I, Red-QAOA [8] takes 16.7s, composed of the classical part and quantum part, and shows an inferior success rate of 0.03%.

Note that the penalty-based QAOA inherently expresses the



(a) The solution space under prior QAOA designs.



(b) The foundation of commute Hamiltonian-based approach.

Fig. 1. Comparison between Choco-Q and other QAOA designs.

Hamiltonian matrix described by the Ising model [16], which is a mathematical representation of the evolution of a quantum system. Therefore, an alternative approach to encode constraints is to develop other types of Hamiltonians, such as the quantum alternating operator ansatz that encodes the constraints into the driver Hamiltonian [9], [17]. However, the prior Hamiltonian-based method has to restrict the constraints in a specific format, lacking the generality to address arbitrary linear equality. For example, the cyclic Hamiltonian-based simulation [9] only supports the equality described in summation format (e.g., $x_1 + x_2 + x_3 = 2$, or $-x_1 - x_2 - x_3 = -1$). Thus, when coping with complex constraints, it may locate solutions in the non-constrained space, as shown in Fig. 1 (a). These approaches also exhibit poor success rates, as shown in Table I. Moreover, the implementation of Hamiltonian requires decomposing it into deployable basic gates, accounting for exponential time complexity, leading to long end-to-end latency.

In this work, we propose Choco-Q, a formal and universal framework that smoothly (like a chocolate) tackles all these limitations. The key novelty of it lies on an expressive representation for encoding the constraints — commute Hamiltonian. In quantum mechanics, Heisenberg's picture states that if the Hamiltonian commutes with an operator, the expectation of the operator remains invariant during the evolution [18]. As illustrated in Fig. 1 (b), the expected value of the operator is calculated by projecting the quantum state onto the operator axis. To maintain a constant expectation value, the quantum states are constrained on the plane orthogonal to the operator axis. With this understanding, we can deduce that if a Hamiltonian commutes with the constraints operator, the states will be restricted to a subspace where each state adheres to the constraints. Starting with the commute Hamiltonian, we comprehensively reformulate the QAOA algorithm flow, which is capable of precisely covering arbitrary linear constraints. Furthermore, restricting the evolution under the commute operator substantially shrinks the search space, bringing a great reduction in the number of iterations.

Then, we present four optimization passes to facilitate end-to-end acceleration and make Choco-Q deployable on current NISQ devices. First, we propose to decouple the commute Hamiltonian into a series of local Hamiltonians, preventing massive tensor computation in the classical part. Unlike the prior Hamiltonian, which repeatedly executes the Hamiltonian for approximation, we only need to execute one block with less circuit depth. Though we serialize the entire Hamiltonian into smaller ones, it still has 4^N degrees of freedom, leaving a huge unitary decomposition overhead for generating high-quality circuits. To handle this, we design a novel unitary decomposition for the commute Hamiltonian, which achieves linear complexity in both time and circuit depth while keeping strict equivalence. Since the noise can make the quantum states outside the constraints, we propose a mid-measurement and post-selection technique to mitigate errors in noisy devices.

The main contributions of this paper include:

- We propose a formal, general, and deployable algorithm for constrained binary optimization problems, which leverages the commute Hamiltonian to encode the constraints. This is the first work that accomplishes the goal of a fast and high-confident solver for this problem.
- We propose a series of optimization techniques to improve the deployability of the commute Hamiltonian on quantum hardware, including Hamiltonian serialization, equivalent decomposition and mid-measurement and post-selection. We demonstrate the effectiveness and reliability of these optimizations by introducing two lemmas, which theoretically ensure the linear complexity in both decomposition time and circuit depth.
- We conduct rigorous experiments on real-world quantum devices using three practical application scenarios: facility location, graph coloring, and K-partition problem. Choco-Q shows remarkable success rates and achieves end-to-end speedup compared to prior works.

At the algorithmic level, for the problems that prior methods can find the optimal solution, Choco-Q increases the success rate by $157\times$ compared to the state-of-the-art QAOA algorithm that applies cyclic Hamiltonian [9]. For the problems that prior methods fail to solve, Choco-Q still exhibits 9.5% - 54% success rates. On real-world quantum hardware, Choco-Q improves the success rate by $2.65\times$ and achieves a $4.69\times$ speedup compared to [9]. Besides, we reduce the decomposition time over $10^6\times$ and reduce the circuit depth more than $10^4\times$, compared to the existing compilation methods [19]. Choco-Q is publicly available on https://github.com/JanusQ/Choco-Q.

II. Background

A. Constrained Binary Optimization

The constrained binary optimization problem aims to find an optimal assignment of binary variables to minimize or maximize the objective function meanwhile satisfying several constraints. The objective function takes the binary variables as inputs and returns a scalar. The constraints are represented using linear equations,

$$\begin{aligned} \min_{\vec{x}} \text{ or } \max_{\vec{x}} \quad & f(\vec{x}), \quad \vec{x} = \{x_1, x_2, \cdots, x_n\} \\ s.t. \quad & C\vec{x} = \vec{c}, \quad \vec{x} \in \{0,1\}^{\otimes n} \end{aligned} \tag{1}$$

where C is the constraint matrix that contains the coefficients of each linear equation. Fig. 2 (a) gives an example with four binary variables and two constraint equations. And the optimal solution to maximize the objective function is $\{1, 0, 1, 0\}$. The time complexity of classical algorithms grows exponentially with the number of variables and constraints, e.g., the worst-case time complexity is $O(2^n)$ using integer linear programming [6].

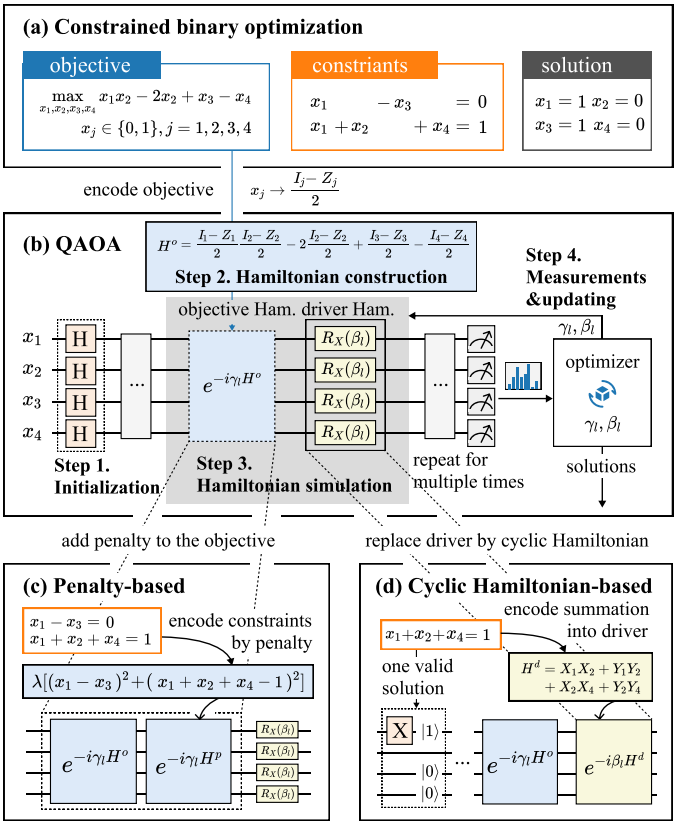


Fig. 2. Solving a constrained binary optimization using QAOA-based designs. (a) The example of a constrained binary optimization problem. (b) A typical QAOA flow encodes objective into objective Hamiltonian without the support of encoding constraints. (c) Extending QAOA by adding penalty terms to objective for the constrained problem. (d) Applying cyclic Hamiltonian to encode constraints (only in summation format) into the driver Hamiltonian H^d.

B. QAOA Preliminaries

For binary optimization, the QAOA algorithm puts each variable in a qubit and prepares a parameterized quantum circuit to encode the objective function. Generally, it consists of four steps, as shown in Fig. 2 (b). **Step 1** initializes the input states through a layer of H gates, which generates a uniform distribution across all possible variable assignments. **Step 2** constructs the Hamiltonian H^o of the objective function by substituting the variable x_j with $\frac{I_j - Z_j}{2}$ in the objective, where I_j and Z_j are the identity operator and the Pauli Z operator on qubit j, respectively. The driver Hamiltonian comprises a set of R_X gates, which is used to parameterize the variable assignments for variational optimization. **Step 3** simulates Hamiltonian by executing the quantum circuit after compilation. This circuit involves multiple layers of unitary $e^{-i\gamma_l H^o}$ and $R_X(\beta_l)$ gates, where γ_l and β_l are the parameters for the l-th layer. **Step 4** measures all qubits and generates a sequence of bitstrings representing the variable assignments. Specifically, the QAOA algorithm calculates the results of the objective function under these assignments. Then, it iteratively updates the parameters in Hamiltonian simulation to approximate the optimal solution, e.g., using gradient descent and linear approximation method.

A natural idea to support constrained binary optimization is to insert the constraints into the objective function as a penalty term [7] (we call it *soft constraint*). For example, in Fig. 2 (c), the constraints are formulated by introducing a positive coefficient λ.

$$\lambda[(x_1 - x_3)^2 + (x_1 + x_2 + x_4 - 1)^2]$$

Theoretically, to ensure the constraints are satisfied, this penalty term should be zero when converging to the optimal solution. During Hamiltonian simulation, an additional penalty Hamiltonian H^p is required, which can be calculated by substituting $x_j = \frac{I_j - Z_j}{2}$ into the penalty term.

An alternative approach is to design a new driver Hamiltonian for the constraints (we call it *hard constraint*). For example, the cyclic Hamiltonian [9] is able to deal with the summation format of the constraints, which is inspired by the one-dimensional Ising model [16].

$$H^d = \sum_{i=1}^{n-1} X_i X_{(i+1)} + Y_i Y_{(i+1)} \tag{2}$$

Here X_i and Y_i are the Pauli-X and Pauli-Y operators on qubit q_i, respectively, which are determined by the variables in constraint. For example, the constraint $x_1 + x_2 + x_4 = 1$ involves variables x_1, x_2, x_4, and its cyclic Hamiltonian is

$$H^d = X_1X_2 + Y_1Y_2 + X_2X_4 + Y_2Y_4$$

In addition, we need to set the initial state to the solution of the constraint equation. For example, one solution of the constraint equation $x_1 + x_2 + x_4 = 1$ is $x_1 = 1, x_2 = 0, x_4 = 0$, thereby setting the three qubits to $|100\rangle$ as shown in Fig. 2 (d).

III. Choco-Q Formulation

Compared to penalty-based methods, encoding to driver Hamiltonian allows hard constraints and higher accuracy. However, the cyclic Hamiltonian is fundamentally limited by the summation format of the constraints that all coefficients must have the same sign, e.g., $x_1 + x_3 = 1$, $-x_1 - x_2 = -1$, and different constraint equations cannot share the same variables. Such limitation arises from the fact that the cyclic Hamiltonian, derived from the Ising model, has to keep the number of excited states $|1\rangle$ constant to encode the constraint.

Inherently, QAOA is a discrete simulation of the quantum evolution under objective Hamiltonian and driver Hamiltonian. Furthermore, for a linear constraint $\sum_{i=1}^{n} c_i x_i = c$, its operator is defined as,

$$\hat{C} = \sum_{i=1}^{n} c_i \frac{I - \sigma_i^z}{2} \tag{3}$$

where the driver Hamiltonian is responsible for maintaining the expected value of this operator constant during evolution. On the other hand, the Heisenberg picture states that if the driver Hamiltonian commutes with the operator, its expected value of the operator will remain unchanged [18]. With this understanding, we propose to employ the commute operator as driver Hamiltonian, namely commute Hamiltonian, which turns out to be a more general QAOA algorithm that enables arbitrary linear constraints for binary optimization, facilitating accuracy, scalability, and deployability.

A. Applying Commute Hamiltonian to QAOA

The commute Hamiltonian also supports hard constraints that can always satisfy the constraints. Commute Hamiltonian originates from the Heisenberg picture, which is an equal expression

of the Schrödinger equation [20]. The Heisenberg picture gives that for any operator $\hat{A}$, the variation rate $\frac{d\hat{A}}{dt}$ under the evolution of Hamiltonian H is proportional to the commutation operator between $\hat{A}$ and H.

$$\frac{d\hat{A}}{dt} = \frac{i}{\hbar}[\hat{A}, H] \\ [\hat{A}, H] = \hat{A}H - H\hat{A} \tag{4}$$

Here, $[\hat{A}, H]$ is the commutation operator. $[\hat{A}, H] = 0$ leads to $\frac{d\hat{A}}{dt} = 0$, which implies that the expected value of $\hat{A}$ will not change during the evolution of H. We then claim that this commutation condition can make the evolved quantum states satisfy constraints if the initial state satisfies constraints.

Lemma 1. *If the initial state satisfies constraints, the collapsed quantum states under a Hamiltonian H commuting with the constraints operator satisfy constraints.*

Proof. Let $|\Phi(t)\rangle = \sum_{j=1}^{N} a_j(t)\,|x_j\rangle, |x_j\rangle = |x_j^1 x_j^2 \cdots x_j^n\rangle$ be the quantum state at time t, and $|x_0\rangle = |x_0^1 x_0^2 \cdots x_0^n\rangle$ is the initial quantum state that satisfy $\sum_{i=1}^{n} c_i x_0^i = c$. For the constraints operator

$$\hat{C} = \sum_{i=1}^{n} c_i \frac{I - \sigma_i^z}{2} \tag{5}$$

its expected value can be simplified as follows:

$$\langle\Phi(t)|\,\hat{C}\,|\Phi(t)\rangle = \langle\Phi(t)|\,(\hat{C}\,|\Phi(t)\rangle) \tag{6}$$

Since

$$\begin{aligned}\hat{C}\,|\Phi(t)\rangle &= \sum_{i=1}^{n} c_i \frac{I - \sigma_i^z}{2}\Big(\sum_{j=1}^{N} a_j(t)\,|x_j\rangle\Big) \\ &= \sum_{j=1}^{N} a_j(t)\Big(\sum_{i=1}^{n} c_i \frac{I - \sigma_i^z}{2}\,|x_j\rangle\Big) \\ &= \sum_{j=1}^{N} a_j(t)\Big(\sum_{i=1}^{n} c_i x_j^i\,|x_j\rangle\Big)\end{aligned}$$

Equation (6) can be rewritten as,

$$\begin{aligned}\langle\Phi(t)|\,\hat{C}\,|\Phi(t)\rangle &= \Big(\sum_{j=1}^{N} a_j^\dagger(t)\,\langle x_j|\Big)\Big(\sum_{j=1}^{N} a_j(t)\sum_{i=1}^{n} c_i x_j^i\,|x_j\rangle\Big) \\ &= \sum_{j=1}^{N}[a_j^\dagger(t) a_j(t)\sum_{i=1}^{n} c_i x_j^i] \\ &= \sum_{j=1}^{N}[|a_j(t)|^2\sum_{i=1}^{n} c_i x_j^i]\end{aligned} \tag{7}$$

Similarly, for initial quantum state $|x_0\rangle$, its expected value of $\hat{C}$ is

$$\langle x_0|\,\hat{C}\,|x_0\rangle = \sum_{i=1}^{n} c_i x_0^i = c \tag{8}$$

Because the expectation is unchanged under the evolution of commute Hamiltonian,

$$\langle\Phi(t)|\,\hat{C}\,|\Phi(t)\rangle = \langle x_0|\,\hat{C}\,|x_0\rangle$$

Combining with the simplified expression in Equation 7, we have

$$\sum_{j=1}^{N}[|a_j(t)|^2\sum_{i=1}^{n} c_i x_j^i] = c \tag{9}$$

since$\sum_{j=1}^{N}|a_j(t)|^2 = 1$, then

$$\sum_{j=1}^{N}[|a_j(t)|^2\sum_{i=1}^{n} c_i x_j^i] = \sum_{j=1}^{N}[|a_j(t)|^2 \cdot c] \tag{10}$$

the left side of the equation minus the right side yields

$$\sum_{j=1}^{N}[|a_j(t)|^2(\sum_{i=1}^{n} c_i x_j^i - c)] = 0 \tag{11}$$

Because $|a_j(t)|^2$ is changing with time t, to make this equality hold for arbitrary $|a_j(t)|^2$, then

$$\sum_{i=1}^{n} c_i x_j^i - c \begin{cases} = 0, & \text{if} \quad |a_j(t)| \neq 0 \\ \neq 0, & \text{if} \quad |a_j(t)| \equiv 0 \end{cases} \tag{12}$$

This means any allowed quantum state $|x_j\rangle$ ($|a_j(t)| \neq 0$) must satisfy the constraints. □

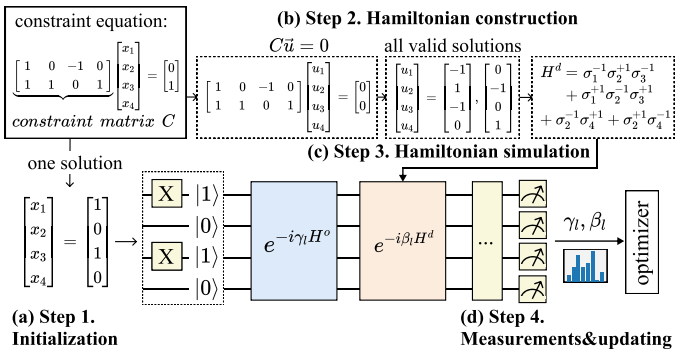


Fig. 3. The workflow of encoding the constraints via commute Hamiltonian.

Then, we need to find a Hamiltonian H that commutes with the constraint operator $\hat{C}$ for constrained binary optimization, where $[\hat{C}, H] = 0$. In QAOA, the Hamiltonian H comprises the objective Hamiltonian H^o and the driver Hamiltonian H^d, denoted as $H = H^o + H^d$. Note that arbitrary objective Hamiltonian H^o commutes with $\hat{C}$ as H^o only involves I and σ^z operators, which always commute with the σ^z operator in $\hat{C}$.

Next, our goal is to find the driver Hamiltonian H^d that also commutes with $\hat{C}$. Hannes et al. [21] gives a universal equation to find the commute Hamiltonian for specific constraints,

$$\begin{aligned} H^d &= \sum_{\vec{u}\in\Delta} H_c(\vec{u}) = \sum_{\vec{u}\in\Delta}(\sigma_1^u \cdots \sigma_n^u + \sigma_1^{-u} \cdots \sigma_n^{-u}) \\ \vec{u} &= \{u_1, u_2, u_3, ..., u_n\}, \quad u_i \in \{1, 0, -1\} \\ \sigma_i^{+1} &= \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad \sigma_i^0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \sigma_i^{-1} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \end{aligned} \tag{13}$$

where Δ is the set of all valid solutions of $\vec{u}$ that satisfies $C\vec{u} = 0$, and $H_c(\vec{u})$ is the commute Hamiltonian for each valid $\vec{u}$.

Fig. 3 depicts the basic flow of applying commute Hamiltonian to QAOA. Similarly, the initial quantum states are set to one solution of the constraint equation. The main difference lies in the design of the commute Hamiltonian and its simulation. Since the constraints are encoded by setting the commutation operator to zero $[\hat{C}, H^d] = 0$, all solutions of $C\vec{u} = 0$ to form the commute Hamiltonian. For example, $\vec{u}^1 = [-1, 1, -1, 0]$ and $\vec{u}^2 = [0, -1, 0, 1]$ are all valid solutions for the case in Fig. 3 (b), resulting in the following Hamiltonian according to Equation (13).

$$\begin{aligned} H^d &= H_c(\vec{u}^1) + H_c(\vec{u}^2) \\ H_c(\vec{u}^1) &= \sigma_1^{-1}\sigma_2^{+1}\sigma_3^{-1} + \sigma_1^{+1}\sigma_2^{-1}\sigma_3^{+1} \\ H_c(\vec{u}^2) &= \sigma_2^{-1}\sigma_4^{+1} + \sigma_2^{+1}\sigma_4^{-1} \end{aligned} \tag{14}$$

Consequently, the Hamiltonian simulation process is formulated as repeatedly executing the objective Hamiltonian and the driver Hamiltonian.

$$\begin{aligned} |\phi_\theta\rangle &= \underbrace{e^{-i\beta H} e^{-i\gamma H} \cdots e^{-i\beta H} e^{-i\gamma H}}_{\text{repeat for } L \text{ times}} |\vec{x^*}\rangle \\ \theta &= \{\gamma_l, \beta_l\}_{l=1}^{L} = \{\beta_1, \gamma_1, \beta_2, \gamma_2, ..., \beta_L, \gamma_L\} \end{aligned} \tag{15}$$

Here, $|\vec{x^*}\rangle$ is the initial state in step 1 (e.g, it is $|101\rangle$ Fig. 3 (a)), $|\phi_\theta\rangle$ is the final state, and θ denotes $2L$ adjustable parameters for approximate optimization. In this manner, the cyclic Hamiltonian can be regarded as a special case of the commute Hamiltonian ($c_i \equiv 1$ in Equation (3)).

B. Advantages and Challenges

Applying commute Hamiltonian is a much more general encoding scheme that deals with arbitrary linear constraints. On the other hand, by restricting the evolution under commute operator, the search space is significantly shrunk, resulting in the high quality of the solution. More importantly, compared to the penalty-based QAOA that often requires all-to-all entanglement between qubits, commute Hamiltonian exhibits sparse connectivity, making it hardware-friendly.

Though the commute Hamiltonian shows higher generality and preciseness, it inevitably introduces computational overhead to obtain the accurate Hamiltonian. According to Equation (13), the calculation of the commute Hamiltonian involves massive tensor-product and accumulation, further aggravating the overwhelming computational pressure. For n qubits Hamiltonian, the space complexity of calculation is $O(2^n)$ and the time complexity is $O(n2^n)$.

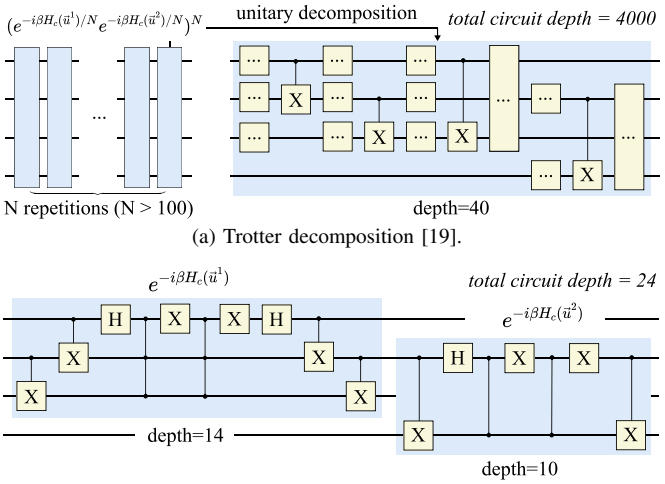
Like other QAOA approaches, the Hamiltonian unitary usually leads to huge circuit complexity after decomposition, making it challenging to deploy on NISQ devices. Typically, existing decomposition methods show exponential complexity to approximate the unitary [22], and often exhibit high approximation error [23]. For example, trotter decomposition [19] approximates the unitary $e^{-i\beta H^d}$ by dividing it into a series of small unitaries $e^{-i\beta H^d/N}$.

$$e^{-i\beta H} = \underbrace{e^{-i\beta H/N}\cdots e^{-i\beta H/N}}_{\text{repeat for N times}} \tag{16}$$

However, the accuracy of trotter decomposition depends on the number of $e^{-iH^d\beta/N}$ repetitions (the error is $O(1/N^2)$ when repeating N times). Moreover, these small unitaries still have to be decomposed into basic gates, severely increasing the circuit complexity, and making it undeployable.

IV. Choco-Q Optimization

To address the new challenges introduced by commute Hamiltonian, we propose multiple optimization passes. To deal with the overhead of tremendous classical computation when calculating the driver Hamiltonian, we serialize the commute Hamiltonian into a set of local Hamiltonian. Unlike the conventional simulation method that exhaustively partitions the Hamiltonian and repeatedly executes them, our serialization technique only needs to calculate the local Hamiltonian that takes less computational cost (Section IV-A). Each local commute Hamiltonian still requires to be decomposed into basic gates for deployment. Different from prior approximation-based unitary decompositions [19], [22], [23], we present an equivalent transformation that decomposes the commute Hamiltonian into several control gates and phase gates. Using this, we demonstrate the linear complexity in both decomposition time and circuit depth (Section IV-B). By putting these two techniques together, we effectively prevent around GFlops tensor computation and reduce the circuit depth from 10^{10} to $\sim$1000. To further improve deployability on the current NISQ devices and make the circuit more error tolerant, we identify



(a) Trotter decomposition [19].

(b) Serialization of commute Hamiltonian.

Fig. 4. Different Hamiltonian simulation methods for Equation (14).

that the constraints satisfaction is disrupted between layers. We then insert mid-measurements between circuit layers to reduce the executed circuit depth and leverage post-selection to mitigate the propagation of errors (Section IV-C). Finally, we complete a shallow and noise-insensitive circuit, exhibiting high accuracy for solving the constrained binary optimization problem.

A. Serialization of Commute Hamiltonian

The commute Hamiltonian in Equation (13) exhibits highly-entangled complexity that requires connecting a large number of qubits, making it hard to deploy on sparsely-connected quantum chips. A natural idea is to reduce it into a series of smaller Hamiltonian, namely *local Hamiltonian* that acts only on a few qubits. However, note that when H_1 and H_2 are complex matrices, $e^{H_1+H_2} \neq e^{H_1}e^{H_2}$, which means that each item $e^{-i\beta H^d}$ in the driver Hamiltonian (Equation (15)) cannot be directly separated.

$$e^{-i\beta H^d} = e^{-i\beta\sum_{\vec{u}\in\Delta} H_c(\vec{u})} \neq \Pi_{\vec{u}\in\Delta} e^{-i\beta H_c(\vec{u})}$$

Taking $\vec{u}^1 = [-1,0], \vec{u}^2 = [-1,1], \beta = 0.8$ as an example, we can easily verified that $e^{-i\beta(H_c(\vec{u}^1)+H_c(\vec{u}^2)} \neq e^{-i\beta H_c(\vec{u}^1)}e^{-i\beta H_c(\vec{u}^2)}$. Essentially, the function of the commute Hamiltonian lies on the encoding of the constraints. In other words, though the separation of Hamiltonian fails to keep the equivalence, we prove that it still meets the constraint operator of Equation (3), which provides the opportunity to shrink the circuit complexity.

Lemma 2. *Replacing the commute Hamiltonian unitary $e^{-i\beta H^d}$ with the serialization of unitaries $\Pi_{\vec{u}\in\Delta}e^{-i\beta H_c(\vec{u})}$ still satisfies the constraint operator during evolution.*

$$|x_c\rangle = e^{-i\beta H}|x\rangle,\ |x_s\rangle = \Pi_{\vec{u}\in\Delta}e^{-i\beta H(\vec{u})}|x\rangle$$
$$\langle x_s|\hat{C}|x_s\rangle = \langle x_c|\hat{C}|x_c\rangle$$

Proof. The expected value of the constraints operator $\hat{C}$ is calculated by the inner product $\langle x|\hat{C}|x\rangle$, where $|x\rangle$ is the initial state. The quantum state $|x_c\rangle$ after applying the commute Hamiltonian unitary satisfies the constraint during evolution:

$$\langle x_c|\hat{C}|x_c\rangle = \langle x|\hat{C}|x\rangle \tag{17}$$

Then, we use $\hat{B}$ to denote $\Pi_{\vec{u}\in\Delta}e^{-i\beta H_c(\vec{u})}$. The expected value of $\hat{C}$ after applying this serialized unitaries $\hat{B}$ can be calculated as follows,

$$\langle x_s|\hat{C}|x_s\rangle = \langle \hat{B}x|\ \hat{C}\hat{B}|x\rangle = \langle x|\hat{B}^\dagger\ \hat{C}\hat{B}|x\rangle \tag{18}$$

where $\hat{B}^\dagger$ is the conjugate of $\hat{B}$. On the other hand, we can easily verify that each Hamiltonian $H_c(\vec{u})$ commutes with $\hat{C}$.

$$[H_c(\vec{u}), \hat{C}] = 0$$

Furthermore, since the Hamiltonian unitary can be expanded by its Taylor series,

$$e^{-i\beta H_c(\vec{u})} = \sum_{k=1}^{\infty}(-i\beta H_c(\vec{u})/k!)^k$$

and we have proven that each term $H_c(\vec{u})$ all commute with $\hat{C}$, the Hamiltonian unitaries then commute with $\hat{C}$. Thus, their product also commutes with $\hat{C}$, which implies $\hat{C}\hat{B} = \hat{B}\hat{C}$. Substituting it into Equation (18), we have

$$\langle x_s|\hat{C}|x_s\rangle = \langle x_c|\hat{C}|x_c\rangle \qquad \square$$

Lemma 2 gives the theoretical foundation for the serialization of commute Hamiltonian, which is specific to the constraint binary optimization problem. Compared to conventional Hamiltonian simulation methods [24], our approach significantly reduces the circuit depth from thousands to dozens, making it possible to implement the circuit. For example, Fig. 4 (a) shows trotter decomposition [19] that is widely used for implementing the driver Hamiltonian (Equation (16)). This method cuts the Hamiltonian into N pieces and repeatedly executes them, leading to tremendous gates and huge circuit depth. In contrast, Fig. 4 (b) depicts the circuit that embeds the constraints using the serialization of commute Hamiltonian. We effectively eliminate the repetition of massive small Hamiltonian, reducing the circuit depth from 4000 to 24 (after decomposition). Moreover, our method exhibits less connection between qubits, which is more hardware-friendly when compiling the circuit for a certain topology. For instance, $H_c(\vec{u}^1)$ acts on three qubits q_1, q_2, q_3, and $H_c(\vec{u}^2)$ only involves two qubits q_2, q_4.

B. Decomposition of Commute Hamiltonian

Though we serialize the large commute Hamiltonian into a set of smaller Hamiltonian, it still requires decomposing these unitaries into basic gates. In particular, the decomposition quality highly determines the final accuracy and the deployability, necessitating a precise and fast methodology. To achieve this, we identify that applying the local commute Hamiltonian to its eigenstate fundamentally behaves like a phase gate,

$$\begin{aligned} e^{-i\beta H(\vec{u})}|x^\pm\rangle &= e^{\mp i\beta}|x^\pm\rangle \\ e^{-i\beta H(\vec{u})}|other\rangle &= |other\rangle \end{aligned} \tag{19}$$

where $|x^\pm\rangle$ is the eigenstate.

$$\begin{aligned} |x^\pm\rangle &= \frac{|v_1\cdots v_n\rangle \pm |\bar{v}_1\cdots\bar{v}_n\rangle}{\sqrt{2}} \\ v = \frac{1+u_i}{2}, \bar{v}_i &= \frac{1-u_i}{2} \end{aligned} \tag{20}$$

Inspired by this property, we first introduce an equivalent transformation that represents the Hamiltonian into multi-phase control gates. Then, we demonstrate the linear complexity in both time and circuit depth.

Algorithm 1: Implementation of converting gates G

Input: solution $\vec{u} = \{u_1, u_2, \cdots, u_n\}$ of $C\vec{u} = 0$.
Output: the circuit to implement G gates in Equation (22).
1: **for** $i = 1$ to n **do**
2: $v_i = (1 + u_i)/2$. // Equation 20
3: **end for**
4: // turn last n-1 qubits into $|11\cdots1\rangle$
5: **for** $i = n$ to 2 **do**
6: CX with target qubit q_i and control qubit q_{i-1}.
7: **if** $v_i = v_{i-1}$ **then**
8: applying X on qubit q_i.
9: **end if**
10: **end for**
11: // current state is $|s^\pm\rangle = (|0\rangle \pm |1\rangle)|1\cdots1\rangle$
12: applying H on the first qubit.// $|s^{+(-)}\rangle \to |0(1)1\cdots1\rangle$

Lemma 3. *Each commute Hamiltonian $e^{-i\beta H_c(\vec{u})}$ can be equivalently decomposed as follows.*

$$e^{-i\beta H(\vec{u})} = G^\dagger P(\beta)X_1P(-\beta)X_1G \tag{21}$$

Here, X_1 gate refers to an X gate applied on the first qubit q_1, G is a set of gates to convert the eigenstate $|x^+\rangle$ and $|x^-\rangle$ into basis state $|01\cdots1\rangle$ and $|11\cdots1\rangle$, respectively.

$$G|x^+\rangle = |01\cdots1\rangle, G|x^-\rangle = |11\cdots1\rangle \tag{22}$$

$P(\beta)$ is a multi-control phase gate that only adds a phase $e^{i\beta}$ to the state $|1\cdots1\rangle$ and keeps the other eigenstates as same.

$$\begin{aligned} P(\beta)|1\cdots1\rangle &= e^{i\beta}|1\cdots1\rangle \\ P(\beta)|other\rangle &= |other\rangle \end{aligned} \tag{23}$$

Proof. We prove this lemma by demonstrating that the decomposition in Equation (21) maintains the same eigenstates $|x^\pm\rangle$. According to Equation (22), taking $|x^+\rangle$ as an example,

$$\begin{aligned} &G^\dagger P(\beta)X_1P(-\beta)X_1G|x^+\rangle \\ =&G^\dagger P(\beta)X_1P(-\beta)X_1|01\cdots1\rangle \end{aligned}$$

applying X_1 gate on qubit q_1 will change $|01\cdots1\rangle$ to $|11\cdots1\rangle$. According to Equation (23), the phase gate $P(-\beta)$ will put a phase $e^{-i\beta}$ on this state.

$$P(-\beta)X_1|01\cdots1\rangle = e^{-i\beta}|11\cdots1\rangle$$

Similarly, according to the definition of $G^\dagger$ and $P(-\beta)$, we can verify that,

$$G^\dagger P(\beta)X_1P(-\beta)X_1G|x^+\rangle = e^{i\beta}|x^+\rangle$$

Furthermore, since the other states cannot activate the multi-control phase gate this decomposition will keep the other eigenstates unchanged.

$$G^\dagger P(\beta)X_1P(-\beta)X_1G|other\rangle = |other\rangle \qquad \square$$

After decomposing the commute Hamiltonian into several convert gates G and phase gates $P(\beta)$, we then illustrate that compiling these gates to basic gates (e.g., Hadamard, Rz, CX) only takes linear time complexity and linear circuit depth. The implementation of the convert gate is shown in Algorithm 1. First, the eigenstate $|x^\pm\rangle$ is calculated using v in Equation 20 (line 1-3). The core idea is to transform the last $n-1$ qubits to $|1\rangle$ state (line 4), which is implemented by CX and X gates (lines 5-10). Clearly, the CX gate helps to flip the target qubit if the control

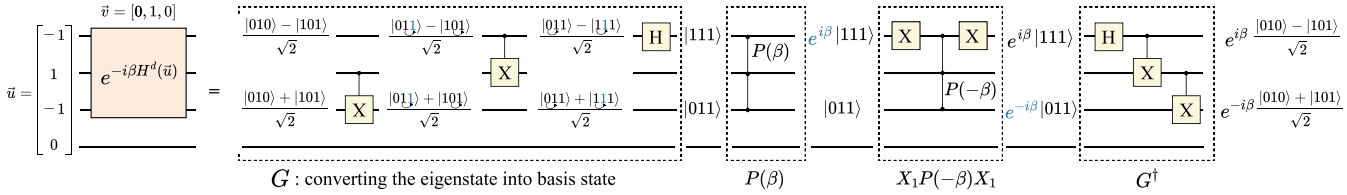


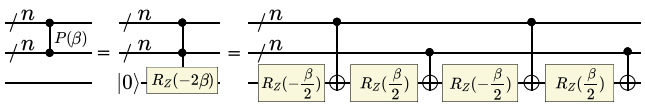
Fig. 5. The end-to-end decomposition flow of the commute Hamiltonian $e^{-i\beta H(\vec{u})}$, following the same example in Equation 14.

qubit is $|1\rangle$, while the X gate flips the state when two qubits are in the same state. After this, the quantum state is manipulated to a reduced state $|s^{\pm}\rangle$.

$$|s^{\pm}\rangle = (|0\rangle \pm |1\rangle)\,|1\cdots1\rangle\,/\sqrt{2} \tag{24}$$

Finally, we use an H gate on the first qubits to get $|11\cdots1\rangle$ for $|x^-\rangle$ and $|01\cdots1\rangle$ for $|x^+\rangle$, respectively. Overall, according to Algorithm 1, the time complexity and circuit depth are with $O(n)$ complexity that is linear to the number of qubits.

The decomposition of the multi-quit phase gate can be visualized as the following equation. We first reformulate the phase



gate $P(\beta)$ as an Rz gate controlled by $2n$ qubits with an ancillary qubit in $|0\rangle$. Next, the control-Rz gate, with $2n+1$ qubits, can be further implemented by four control-X gates and four Rz gates, with each control-X gate involving $n+1$ qubits. Since the control-X gates can be implemented with $16n$-qubit gate with one ancillary qubit [25], the overall complexity of decomposing the n-qubit phase gate is $O(n)$. On the other hand, there are two ancillary qubits in total, which can be reused in the entire circuit of commute Hamiltonian simulation. Thus, the overall circuit complexity is also $O(n)$ with only two ancillary qubits.

Unlike prior approximation-based unitary decomposition [19], [22], [23], we design a precise decomposition formulation for implementing the commute Hamiltonian with linear complexity. Instead of numerically calculating the Hamiltonian through tensor computations in Equation (13), we can directly derive the decomposed circuit from the solution vector $\vec{u}$. Following the example $H_c(\vec{u}^1)$ in Equation (14), we can calculate the eigenstate according to Equation (20).

$$\vec{u}^1 = [-1, 1, -1, 0]$$
$$\text{Eq (12)} \Rightarrow \quad |x^{\pm}\rangle = (|010\rangle \pm |101\rangle)/\sqrt{2}$$

Fig. 5 provides the end-to-end decomposition flow for this Hamiltonian. The G gates are constructed by two CX gates and an H gate. Since q_2 and q_3 are not in the same state, the first CX gate sets the qubit q_3 into $|1\rangle$. Similarly, the second CX gate turns qubit q_2 into $|1\rangle$. Then, the H gate on qubit q_1 changes the state $(|0\rangle + |1\rangle)\,|11\rangle\,/\sqrt{2}$ to $|011\rangle$ and changes the state $(|0\rangle - |1\rangle)\,|11\rangle\,/\sqrt{2}$ to $|111\rangle$. The phase gate $P(\beta)$ puts phase $e^{i\beta}$ to state $|111\rangle$, and $X_1P(-\beta)X_1$ puts $e^{-i\beta}$ to state $|011\rangle$. After applying the inversion gate $G^\dagger$, we successfully add phase $e^{i\mp\beta}$ on state $|x^{\pm}\rangle$, constructing the whole circuit for the Hamiltonian $H_c(\vec{u}^1)$. Note that we also draw the resultant circuit of the overall driver Hamiltonian in Figure 4, illustrating the linear complexity of circuit depth.

C. Mid-measurement and Post-selection for error mitigation

Although the commute Hamiltonian can maintain the quantum states under constraints theoretically, noise in quantum systems often disrupts the commute condition. A non-commuting Hamiltonian will introduce more invalid states outside the constraints, reducing the in-constraints rate, as illustrated in Lemma 4.

Lemma 4. *In depolarizing noise channel with gate error rate p, the in-constraints rate scales as $(1-p)^{\#gates} + \#gates(1-p)p/2^{rank(C)}$. Here, $\#gates$ is the number of gates in the circuit, and $rank(C)$ is the rank of constraints matrix C.*

Proof. Let $P(\rho)$ be in-constraints rate of density matrix ρ. By the definition of in-constraints rate, we have

$$P(\rho) = \textstyle\sum_{i=1}^{K} \langle v_i|\rho|v_i\rangle \tag{25}$$

Where $|v_i\rangle, i \in \{1, ...K\}$ is the states satisfying constraints and K is the total number of these states, which equals to $2^{n-rank(C)}$ (n is the number of all variables). Let $U = U_1U_2\cdots U_m$ be the circuit unitary of the commute Hamiltonian and m be the total number of gates. A depolarizing channel with gate error rate p changes the density matrix ρ to $E(\rho)$ after every unitary U_i.

$$E(\rho) = (1-p)\rho + \frac{p}{2^n}I \tag{26}$$

Thus, after applying the unitary gate U with the depolarizing channel, the resulting density matrix is

$$E_m(\rho) = (1-p)^m U\rho U^\dagger + (m(1-p) + o(p^2))p/2^n I \tag{27}$$

Here, $o(p^2)$ is a higher-order polynomial (above linear) of p and negligible. The in-contraints rate of the density matrix $E_m(\rho)$ is

$$\begin{aligned} P(E_m(\rho)) &= \textstyle\sum_{i=1}^{K} \langle v_i|E_m(\rho)|v_i\rangle \\ &= (1-p)^m \textstyle\sum_{i=1}^{K} \langle v_i|U\rho U^\dagger|v_i\rangle + m(1-p)\frac{p}{2^n}K \\ &= (1-p)^m P(U\rho U^\dagger) + m(1-p)p/2^{rank(C)} \end{aligned}$$

Since for the noise-free density matrix, $P(U\rho U^\dagger) = 1$, we can get the in-constraints rate after applying $\#gates$ gates is

$$(1-p)^{\#gates} + \#gates \cdot (1-p) \cdot p/2^{rank(C)} \qquad \square$$

To mitigate the reduction of in-constraints rate, we propose *Mid-Measurement and Post-Selection (MMPS)* optimization. In QAOAs, parameter updating is guided by the probability distribution obtained from the previous iteration. This insight allows us to reinterpret the circuit as a sequence of layers, where probability information is propagated and updated at each layer. In our commute Hamiltonian-based QAOA, each layer is composed of the objective Hamiltonian simulation and the commute Hamiltonian simulation, as shown in Figure 6 (a).

Different from executing all layers completely in every shot, MMPS operates by inserting mid-measurements after each layer. We then apply post-selection to filter out infeasible solutions and reconstruct an approximate probability distribution that fully

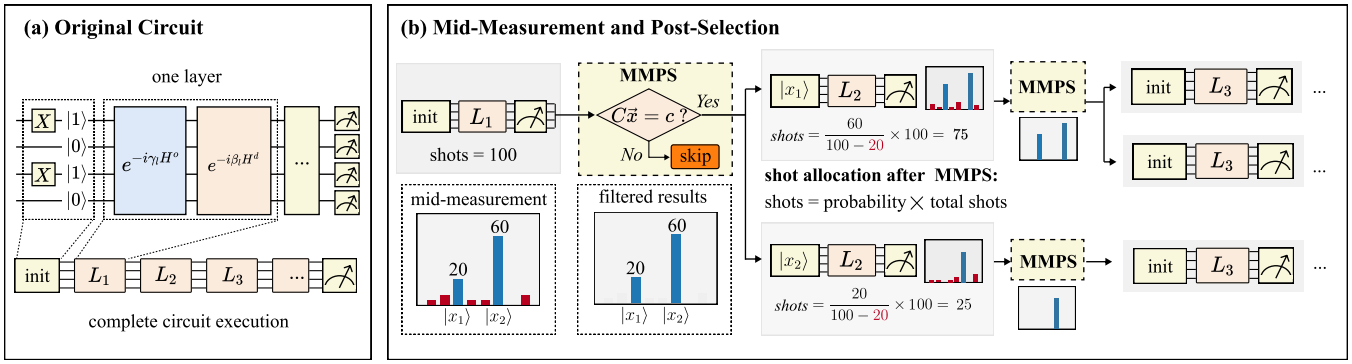


Fig. 6. Using mid-measurement and post-selection (MMPS) to filter out the states outside the constraints for improving the solution quality under noise.

satisfies the constraints. This purified distribution is then used to initialize the next layer, ensuring that the propagation of probability information across the circuit remains consistent and meaningful for parameter optimization.

For example, as illustrated in Fig. 6 (b), we first execute the first layer with 100 shots and use mid-measurements to get the solution distribution of this layer. We can see that 20 states violate the constraints, and these states are discarded, resulting in 60 instances of $|x_1\rangle$ and 20 instances of $|x_2\rangle$. Subsequently, the post-selected probability distribution is used to determine the number of shots for executing the next layer. For instance, the number of shots assigned to input $|x_1\rangle$ is given by $\frac{60}{100-20} \times 100 = 75$.

Although post-selection is performed after each mid-circuit measurement, it has little computational overhead. The verification of the equation $C\vec{x} = \vec{c}$ only involves a simple matrix-vector multiplication under binary vectors, which can be efficiently computed. In our test, for a graph coloring problem with 15 qubits, the total processing time is approximately 7 seconds, while all post-selections take only 0.04 milliseconds.

Furthermore, since the noise can reduce the probability of valid states, this technique requires more shots to maintain the number of measured valid states. Here we give a quantified shots amplification strategy based on Lemma 4. To ensure the number of valid states in the final measurements equals that in a noise-free environment, we derived that the total shots need to be amplified by $\sim 1+pG_L\times$. Here, G_L is the number of gates in a layer since we insert mid-measurements after each layer and only need to amplify the state number of the mid-measurement. In the quantum device with 0.5% gate error rates, every 200 additional gates increase the circuit's total shots by the original shots.

V. Choco-Q: Complexity Analysis

Table II gives the complexity of classical solvers and quantum solvers and provides the approximation quality, quantified by the approximation ratio gap (ARG).

$$ARG = |\frac{E(f(\vec{x}) + \lambda\|C\vec{x} - \vec{c}\|)}{f(\vec{x}_{optimal})} - 1| \tag{28}$$

Here, $\vec{x}$ is the outcome of the solver, $\vec{x}_{optimal}$ is the optimal solution, f is the objective function, $C\vec{x} - \vec{c}$ is the constraints equation. λ is the penalty term, which is set to 10 here to balance the objective and constraints. E means the expectation value on all outcomes $\vec{x}$. For the classical solver, the complexity and approximation quality are collected from prior works [26]–[30]. For quantum solvers, since the theoretical approximation quality of QAOA is hard to calculate, we get it by averaging the data

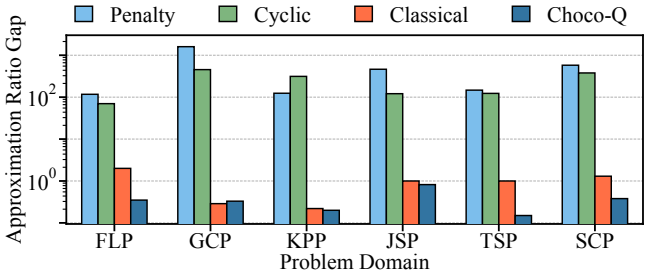


Fig. 7. Approximation ratio gap among different quantum and classical solvers.

from the experiments. For the complexity of QAOA, we divide it into the classical part (classical complexity) and the quantum part (the number of resulting unitaries).

First, we consider a universal problem of constrained binary optimization: let n be the number of variables, and m be the number of constraints. For QAOAs, let p be the number of layers, and S be the shots of measurement.

Classical complexity of Choco-Q. The complexity of solving the equation $C\vec{u} = 0$ is $O(mn^2)$. The complexity of compiling the solution vectors into the commute Hamiltonian is $O((n-m)n)$ since the number of solution vectors $\vec{u}$ is $n-m$ and the compilation of a single commute Hamiltonian takes $O(n)$. The complexity of compiling the objective Hamiltonian is $o(n)$. Therefore, the total classical complexity of Choco-Q is $O(mn^2)$ because $n > m$.

Quantum complexity of Choco-Q. For each sub-problem, the number of iterations is polynomial to the number of parameters $2p$ [31]. In each iteration, the circuit depth is $pO(n+(n-m)d)$ with p layers of objective Hamiltonian $O(n)$ and commute Hamiltonian $O((n-m)d)$, where d is the number of nonzero elements in $\vec{u}$ and $d \ll n-m$. Hence, the total quantum complexity of Choco-Q is $poly(p) \cdot p \cdot O(n) = O(poly(p)n)$.

The complexity of other QAOAs. For the complexity of other QAOAs, we also divide it into classical and quantum parts. For penalty-term-based QAOA [7], its classical complexity originates from the compilation of objective Hamiltonian, which has n objective terms and $mn(n-1)/2$ penalty terms. Thereby, its classical complexity is $O(mn^2)$. For quantum parts, its circuit depth for objective Hamiltonian is also $O(mn^2)$, and the total quantum complexity is $O(poly(p)mn^2)$. For cyclic Hamiltonian-based QAOA [9], the classical complexity is $O((m-g)n^2+n)$ where g is the number of constraints in summation format. Quantum complexity is $O(poly(p)((m-g)n^2+n))$ with $O((m-g)n^2)$

TABLE II
COMPUTATIONAL COST OF CLASSICAL SOLVER AND QUANTUM SOLVER

Application			Universal case	FLP	GCP	KPP	JSP	TSP	SCP
#Variable			n variables	$2fd+f$	v^2+ve	vb	$js+c$	$\sim 2^c$	$\sim 3e$
#Constraint			m constraints	$fd-d+f$	$ve+v$	$v+b$	$j+s$	2^{c-1}	s
Classical Solver	Primal-Dual Greedy [26]		-	$O(fd\log(fd))$	-	-	-	-	-
	DSATUR algorithm [27]		-	-	$O(v^2)$	-	-	-	-
	First Fit Decreasing [28]		-	-	-	$O(v\log v+vb)$	-	-	-
	List Scheduling [29]		-	-	-	-	$O(j\log j)$	-	-
	Greedy Approximation [30]		-	-	-	-	-	$O(c^2)$	$O(se)$
Quantum Solver	**Penalty** [7]	Classical	$O(mn^2)$	$O(f^3d^3)$	$O(v^3e^2)$	$O(v^3b^2)$	$O(j^3s^2)$	$O(2^c)$	$O(se^2)$
		Quantum	$O(Poly(p)Smn^2)$	$O(f^3d^3)$	$O(v^3e^2)$	$O(v^3b^2)$	$O(j^3s^2)$	$O(2^c)$	$O(se^2)$
	Cyclic [9]	Classical	$O((m-g)n^2)$	$O(f^3d^3)$	$O(v^3e^2)$	$O(v^2b^3)$	$O(j^2s^3)$	$O(2^c)$	$O(se^2)$
		Quantum	$O(Poly(p)S(m-g)n^2)$	$O(f^3d^3)$	$O(v^3e^2)$	$O(v^2b^3)$	$O((j^2s^3))$	$O(2^c)$	$O(se^2)$
	Choco-Q	Classical	$O(mn^2)$	$O(f^3d^3)$	$O(v^3e^2)$	$O(v^3b^2)$	$O(j^3s^2)$	$O(2^c)$	$O(se^2)$
		Quantum	$O(Poly(p)Sn)$	$O(fd)$	$O(ve)$	$O(vb)$	$O(js)$	$O(2^c)$	$O(e)$

We omit $Poly(p)S$ factor in the quantum complexity in the specific problems. The description of each problem is as follows:

FLP: Minimize the cost of supplying d demands with f facilities ($f < d$).

GCP: Use minimum colors to color a graph with v vertexes and e edges and each adjacent vertex has a different color ($v < e$).

KPP: Partition the graph with v vertexes and e edges into b blocks and minimum the cut edges between blocks ($b < v < e$).

JSP: Schedule j jobs on s machines to maximizing the total profit. Each machine can accommodate c jobs ($c < s < j$).

TSP: Find the shortest path to visit c cities and each city can be visited exactly once.

SCP: Given s sets, find the minimum sets to cover all e elements.

objective Hamiltonian depth and $O(n)$ cyclic Hamiltonian depth.

In Table II, for specific benchmarks, we also formulate the complexity of quantum solvers using the related variables of the problem. Choco-Q has a similar classical complexity but a much lower quantum complexity, benefiting from our Hamiltonian serialization and decomposition techniques. We also find that the quantum solver complexity on the TSP problem shows an exponential trend, which originates from the binary optimization formulation of the TSP takes exponential binary variables. Finding a polynomial encoding for this problem will be an important research direction in the future.

Figure 7 plots the approximation ratio gap of quantum and classical solvers. The approximation ratio gap is averaged on the experimental results of benchmarks in Table III. In summary, classical solvers' approximation quality is close to Choco-Q and even slightly lower than Choco-Q in GCP. These classical solvers lack the generality for handling various constraint binary optimization problems. Among QAOA designs, Choco-Q improves the approximation quality (ARG) by 884× compared to other QAOA designs.

TABLE III
THE BENCHMARKS FOR EVALUATION.

Domain	Benchmark	#Case	#Var.	#Con.	Description
FLP [32]	F1: 2F-1D	100	6	3	2F-1D means supplying one Demands with two Facilities.
	F2: 3F-2D	100	15	8	
	F3: 3F-3D	100	21	12	
	F4: 4F-3D	100	28	15	
GCP [15]	G1: 3V-1E	100	12	6	3V-1E means the graph has three Vertexes and one Edges.
	G2: 3V-2E	100	15	9	
	G3: 4V-2E	100	24	12	
	G4: 4V-3E	100	28	16	
KPP [33]	K1: 4V-3E-2B	100	8	6	4V-3E-2B means the graph with four Vertexes and three Edges is partitioned into two Blocks.
	K2: 6V-5E-3B	100	18	9	
	K3: 8V-7E-3B	100	24	11	
	K4: 9V-8E-3B	100	27	12	
JSP [34]	J1: 2J-2M-3C	100	7	4	2J-2M-3C means two Jobs is allocated to two Machines and the machine Capacity is three.
	J2: 3J-3M-5C	100	14	6	
	J3: 3J-4M-6C	100	18	7	
	J4: 4J-5M-7C	100	27	9	
TSP [34]	T1: 4C	100	11	8	4C means four Cities.
	T2: 5C	100	27	16	
SCP [35]	S1: 4S-4E	100	9	4	4S-4E means covering four Elements from the subsets of four Sets.
	S2: 5S-5E	100	12	5	
	S3: 6S-6E	100	20	6	

VI. EVALUATION

A. Experiment Setup

Benchmark. We evaluate Choco-Q using seven practical application scenarios, including facility location problem (FLP) [1], graph coloring problem (GCP) [15], k-partition problems (KPP) [33], capital budgeting problem (CBP) [36], job scheduling problem (JSP) [34], set cover problem (SCP) [35], and travelling salesman problem (TSP) [37].Table III lists the setting of these applicationsA. For each application, we collect 400 cases from the related literature [1], [15], [33]. And we categorize these cases into four problem scales (e.g., F1 to F4 in FLP), with the number of variables ranging from 6 to 28, and the number of constraints ranging from 3 to 15.

Comparison. We compare Choco-Q with previous QAOAs that support constrained binary optimization, including Penalty-based QAOA [7], cyclic Hamiltonian-based QAOA [9], and Hardware-efficient ansatz (HEA) [38]. For Penalty-based QAOA, We integrate it with two state-of-the-art QAOA optimization techniques, *FrozenQubits* [12] and *Red-QAOA* [8]. Specifically, *FrozenQubits* aims to reduce the circuit depth and enhance the success rate, while *Red-QAOA* optimizes initial parameters. HEA is a universal variational quantum algorithm (non-QAOA), where we use the circuit provided by Kandala et.al [38]. When designing HEA, we introduce a penalty method to make the output satisfy the constraints as much as possible. For parameter updating, we use

TABLE IV
THE CIRCUIT DEPTH, SUCCESS RATE, AND IN-CONSTRAINTS RATE OF DIFFERENT QAOA DESIGNS UNDER 12 BENCHMARKS.

Benchmark		Circuit depth				Success rate (%)				In-constraints rate (%)				Approximation ratio gap (ARG)			
		Penalty [7]	Cyclic [9]	HEA [38]	Choco -Q	Penalty [7]	Cyclic [9]	HEA [38]	Choco -Q	Penalty [7]	Cyclic [9]	HEA [38]	Choco -Q	Penalty [7]	Cyclic [9]	HEA [38]	Choco -Q
FLP [32]	F1	56	91	42	43	3.79	21.4	8.91	**99.8**	22.0	38.7	26.4	**100.0**	39.0	15.3	44.9	**0.16**
	F2	88	99	98	221	0.14	0.09	✗	**54.0**	0.47	1.37	0.10	**100.0**	107	70.9	150	**0.30**
	F3	92	134	147	275	✗	0.03	✗	**30.0**	0.04	0.79	✗	**100.0**	145	94.6	181	**0.50**
	F4	108	162	196	421	✗	✗	✗	**13.3**	✗	0.74	✗	**100.0**	177	97.7	257	**0.43**
GCP [15]	G1	188	230	84	124	0.15	4.74	0.26	**69.7**	0.50	10.6	0.36	**100.0**	698	217	1144	**0.06**
	G2	221	263	105	358	0.03	0.14	0.03	**67.1**	0.07	0.67	0.03	**100.0**	747	621	1727	**0.25**
	G3	654	708	168	586	✗	✗	✗	**17.1**	0.02	0.27	✗	**100.0**	2592	479	3091	**0.47**
	G4	683	737	196	906	✗	✗	✗	**9.50**	✗	✗	✗	**100.0**	2366	501	3834	**0.56**
KPP [33]	K1	115	150	56	79	1.66	38.2	1.04	**86.1**	5.18	84.8	4.46	**100.0**	53.8	32.3	59.4	**0.14**
	K2	202	244	126	324	0.01	14.2	✗	**52.6**	0.05	39.7	0.04	**100.0**	118	37.8	130	**0.18**
	K3	264	306	168	464	0.01	2.59	✗	**21.1**	0.01	31.6	✗	**100.0**	162	24.7	165	**0.24**
	K4	296	338	189	534	✗	0.45	✗	**13.3**	✗	8.23	✗	**100.0**	163	30.5	170	**0.23**
JSP [34]	J1	72	168	49	67	5.15	13.1	3.54	**84.1**	15.7	38.2	9.17	**100.0**	95.5	43.9	86.0	**0.17**
	J2	118	224	98	184	0.12	2.17	0.02	**24.7**	1.68	20.4	0.28	**100.0**	234	108	291	**0.60**
	J3	134	243	126	298	0.02	1.03	✗	**9.78**	0.53	10.0	0.04	**100.0**	350	138	407	**1.24**
	J4	166	291	189	556	✗	✗	✗	**1.47**	0.04	9.12	✗	**100.0**	1167	193	1137	**1.27**
TSP [37]	T1	192	376	77	121	0.44	0.37	0.06	**99.0**	0.96	1.42	0.11	**100.0**	104	95.0	109	**0.02**
	T2	538	828	189	493	✗	✗	✗	**30.7**	✗	✗	✗	**100.0**	190	150	208	**0.27**
SCP [35]	S1	179	198	65	80	4.22	5.53	1.61	**77.6**	11.9	14.2	6.61	**100.0**	262	186	819	**0.21**
	S2	223	250	76	154	0.59	1.24	0.02	**26.1**	1.17	2.43	0.30	**100.0**	621	400	1064	**0.43**
	S3	449	485	129	231	0.22	0.41	✗	**17.9**	0.63	0.92	0.03	**100.0**	850	549	1221	**0.50**
Improv.(×)		-	-	-	**1.34**	-	-	-	**>157**	-	-	-	**>60.0**	-	-	-	**884**

[1] For Penalty-based QAOA [7], we integrate it with two open-sourced optimization techniques, *FrozenQubits* [12] and *Red-QAOA* [8].
[2] The improvement is compared to the cyclic-based Hamiltonian method [9]. ✗ means that the design fails to find the optimal solution for this case.

the constrained optimization by linear approximation method [39] for all designs.

Platform. We conduct several small-scale evaluations on three IBMQ systems, including *Fez* platform with 159-qubit Heron r2, *Sherbrooke* and *Osaka* platform with 127-qubit Eagle r3 type [10]. Since QAOA usually comprises a large amount of CZ gates, *Fez* is QAOA-friendly as it features the CZ gate as the basic gate with 99.7% fidelity. The other two devices only support single-direction ECR gates with 99.3% fidelity, which takes three ECR gates to implement the CZ gate, resulting in a higher error rate. The simulation experiments and the classical part of QAOA are executed on an AMD EPYC 9554 64-core sever with 1.5T SSD memory. The simulation of the quantum circuit is accelerated by one A100 GPU on the server.

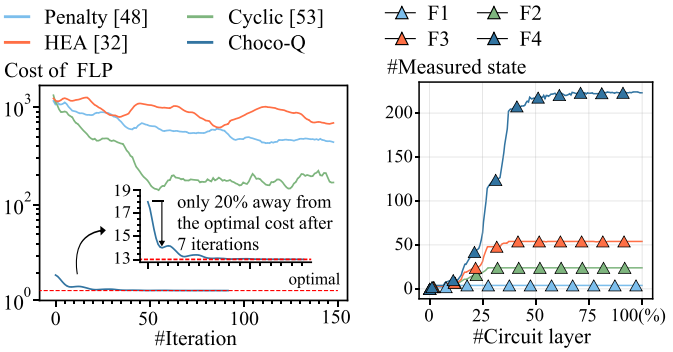
Evaluation metrics. Similar to prior constraint QAOAs, we employ three algorithmic metrics: *success rate*, *in-constraints rate*, and *approximation ratio gap*. The success rate is defined as the probability of getting the optimal solution after measurements. The in-constraints rate is the probability that the output solutions satisfy the constraints. Thus, the in-constraints rate is always higher than the success rate. The approximation ratio gap is defined in Eq (28), which represents the quality of all outputs. When implementing on real-world devices, we further evaluate the end-to-end *run time* (without data communication), including the compilation time for Hamiltonian decomposition, circuit execution time, and the parameter updating time for iterative optimization.

B. Algorithmic Evaluation

Success rate. Compared to other QAOA designs, Choco-Q achieves an average 157× improvement in successfully finding the optimal solution, as shown in Table IV. In particular, for the problem in medium-scale (F2, G2, K2, J2, T1, S2), all other approaches nearly fail to get the optimal solution, with a success rate below 15%. While, our approach exhibits a high success rate, from 52.6% to 67.1%. Even for large-scale problems, we still have a relatively high chance of finding the solution (1.47% - 30.7%). Such great improvement comes from the fact that the commute Hamiltonian effectively narrows down the search space, leading to a higher probability of getting the optimal solution.

In-constraints rate. When encoding the constraints, Choco-Q shows a 100% in-constraints rate, indicating that our algorithm comprehensively takes all constraints into account. This is also the reason that Choco-Q has a much higher success rate. The 100% in-constraints rate benefits from the generality of our encoding scheme that applies commute Hamiltonian to thoroughly represent the constraints. In contrast, the cyclic Hamiltonian restricts constraints to a summation format, making it challenging to consider all constraints, especially in large-scale problems. Besides, the cyclic Hamiltonian performs better on KPP benchmarks since the constraints of KPP are in summation format and involve fewer shared variables. The in-constraints rate of penalty-based QAOA [7] is highly determined by the number of constraints, where each constraint equation corresponds to one penalty term in the objective function. In contrast, Choco-Q is irrelevant to it as we consider the problem as the commute operator between the driver Hamiltonian and the constraints.

Circuit depth. Choco-Q involves a little bit more circuit depth, notably in large-scale problems. First, as mentioned in Section IV-B, prior algorithms are fundamentally approximation-based approaches for optimization problems, which may easily fail to meet the constraints. However, Choco-Q lies on the precise encoding of arbitrary linear constraints, which requires to implementation of the commute Hamiltonian of all solution $\vec{u}$ in Equation 13, resulting in the linear complexity of circuit depth. Taking the G3:4V-2E case as an example, it results in 12 $\vec{u}$ to precisely express the 12 constraint equations, giving rise to large circuit depth. The second reason is that we simulate the other



(a) Convergence using F1:2F-1D. (b) The parallelism of Choco-Q.

Fig. 8. Convergence analysis of Choco-Q.

QAOA algorithms only with seven repeated layers, i.e., repeat seven times in Equation (15). We adopt seven layers because this setting achieves the best trade-off between success rate and circuit depth. We test that even if we increase the number of layers, it only contributes to limited improvement in the success rate. And in Table IV, we do not repeat the driver Hamiltonian of Choco-Q (one layer). In a word, compared to the significant performance improvement, the additional circuit depth is acceptable.

Approximation ratio gap (ARG) indexes the solution quality across all outcomes generated by the algorithm, which is widely used in prior QAOAs [8], [12], [40]. It is defined as Eq (28). Here, the objective function f and constraints C are defined in Equation (1). $\vec{x}$ is the outcome, $x_{\vec{optimal}}$ is the optimal solution, and E means the expectation value on all outcomes. λ is the penalty term, which is set to 10 here to balance the objective and constraints. As shown in Table IV, Choco-Q improves the ARG by 884×. Such improvement originates from that in Choco-Q, $\lambda|C\vec{x}-\vec{c}|$ in Equation (28) is constantly zero, while it can be very huge using other methods. In particular, the ARG is larger in GCP benchmarks since GCP contains more complicated constraints. Even though, the ARG of Choco-Q is below 0.6.

Convergence. Fig. 8 (a) depicts the convergence curves of different QAOA designs, picking one case from the F1:2F-1D benchmark. Choco-Q can reach the optimal cost within 30 iterations, while other methods require over 148 iterations and still have at least a 78% gap with the optimal cost. In particular, Choco-Q can quickly reach 20% away from the optimal cost within 7 iterations, which is 7.3× faster than the cyclic Hamiltonian-based approach. Such high convergence speedup is attributed to a good initial cost (18), while the others are extremely large (10^3). Furthermore, the squeezed search space also helps to find the optimal solution in fewer iterations. Fig. 8 (b) illustrates the parallelism of Choco-Q. We collect the number of measured states through the circuit, which represents the parallelism involved in the superposition of quantum states. Choco-Q shows an exponential growth of the parallelism at the beginning of the circuit, i.e., at the point of around 1/4 circuit. Unlike prior QAOAs that initialize a uniform superposition state, even though Choco-Q prepares a special initial state, we still effectively harvest the quantum parallelism by applying commute Hamiltonian.

C. Evaluation on Real-world Quantum Platforms.

This section evaluates Choco-Q on the IBM cloud quantum devices. Considering the short decoherence time and the inevitable noise error, we choose to implement the small-scale problems, i.e., F1, G1, and K1 cases in Table IV.

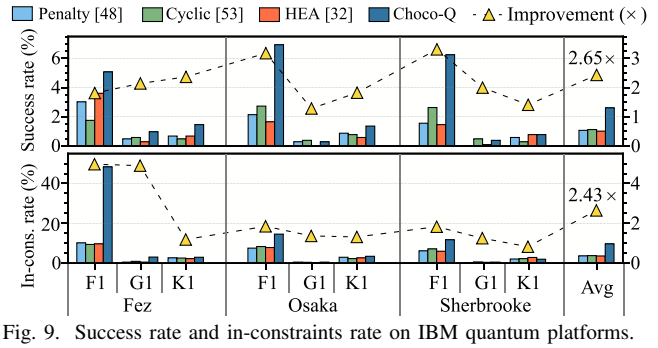
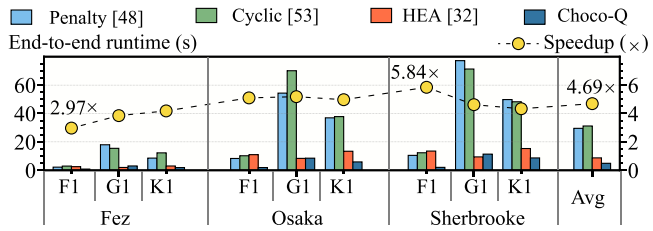
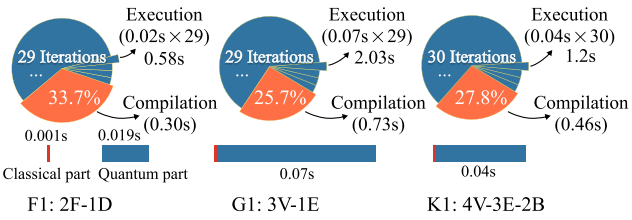


Fig. 9. Success rate and in-constraints rate on IBM quantum platforms.



(a) End-to-end latency comparison on the real-world quantum platforms.



(b) Latency breakdown of Choco-Q on the Fez platform.

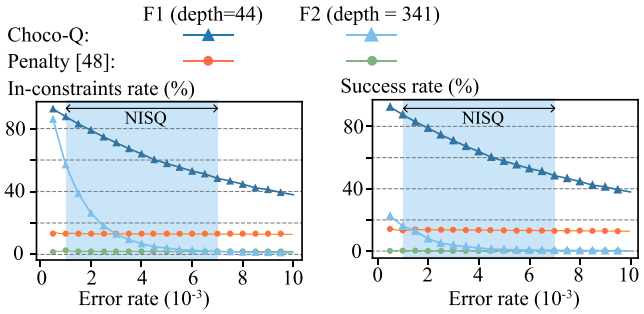
Fig. 10. End-to-end latency evaluation and breakdown.

Success rate. Fig. 9 gives the success rate on three quantum devices. Compared to the noise-free simulator, the success rate of all cases is decreased due to various noise, e.g., cross-talk and hardware defects. Overall, Choco-Q achieves an average of 2.65× improvement over other designs. The effectiveness of NISQ devices arises from our decomposition technique that transforms the commute Hamiltonian into multiple phase gates, which are less sensitive to flip error.

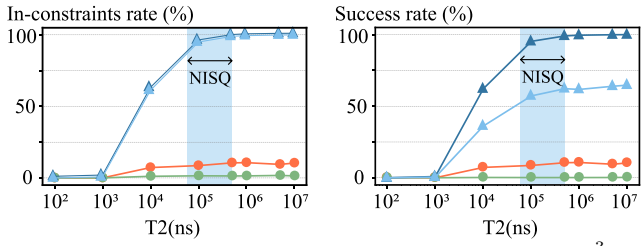
In-constraints rate. Compared to other designs, Choco-Q shows 2.43× higher in-constraints rate. In particular, on the Fez platform, we achieve up to 48% in-constraints rate, resulting in 4.98× improvement, thanks to its architectural properties. As mentioned before, this device is QAOA-friendly, allowing us to fully leverage our technological advantages. As we serialize the driver Hamiltonian into a set of smaller ones that necessitates fewer qubit connections, Choco-Q is more noise-tolerant against the cross-talk noise.

End-to-end latency. Fig. 10 (a) compares the overall latency comprising of the compilation time, circuit execution time, and parameter updating time. Choco-Q achieves 2.97× - 5.84× speedup, and always within 10 seconds, primarily attributed to the reduced number of iterations. The latency of HEA [38], the non-QAOA algorithm, is close to ours in several cases. This is because it shows shallow circuit depth in F1, G1, and K1 cases, leading to around 4× speedup when executing the circuit.

Fig. 10 (b) presents the latency breakdown for the cases in



(a) In-constraints rate and success rate under Pauli noise model.



(b) In-constraints rate and success rate on different T2. T1 is set to $\frac{3}{2}$T2.

Fig. 11. Evaluation under different sources of noise.

the Fez platform. We consider the end-to-end latency as the summation of compilation time (including the decomposition) and execution time. The execution time includes the classical and quantum parts of QAOA, with the classical part only taking a little time. We can see that the iterative execution is the most time-consuming, requiring around 30 iterations and accounting for around 70% of the total latency.

D. Detailed Analysis on Different Noise Model

In this section, we study the performance of Choco-Q without MMPS optimization under different noise models, including the depolarizing and thermal noise models. The depolarizing model definition and its in-constraints rate scaling are given in Lemma 4. In the thermal noise model, we set T1 to $\frac{3}{2}$T2, based on data from the IBMQ Fez device.

Evaluation under depolarizing noise model. Fig. 11 (a) provides the in-constraint rates and success rates under different Pauli error rates. Theoretically, the in-constraint rate scales polynomially as shown in Lemma 4, following the same trend in Fig. 11 (a). The success rate is similar as it highly depends on the in-constraints rate. The advantage of Choco-Q narrows when the error rate increases. This is because the search space of other methods naturally has a much lower probability of successfully finding the solution. The error rate of current NISQ devices typically ranges from 10^{-3} to 7×10^{-3} [10], which is labeled by the blue background box. We can see that the in-constraints rate and success rate of choco-Q on the F1 case still exceed 50%, and are higher than the penalty-based QAOA. We also find that the in-constraints rate on F2 decays faster than F1, since the circuit depth of F2 is almost $8\times$ that of F1. These results align with our analysis in Lemma 4.

Evaluation under thermal noise model. Fig. 11 (b) presents the in-constraint and success rates under different T2 and higher T2 time means smaller thermal noise strength. When T2 increases from 10^3ns to 10^6ns, the in-constraint rate goes to 100%. In NISQ devices, the in-constraints rate is above 80%, indicating that the thermal noise can not significantly degrade the performance

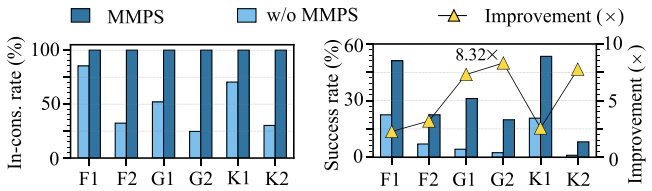


Fig. 12. In constraints rate and success rate after post-selecting the state with constraints by mid-measurements.

Choco-Q circuit. Besides, we can see that the in-constraint rate of Choco-Q is irrelevant to the problem scale. Conversely, for penalty-based QAOA, the in-constraints rate is reduced by over $5\times$ from F1 to F2. As for success rate, Choco-Q achieves an acceptable performance when T2 reaches 10^4ns. Considering the T2 of current NISQ devices (86μs to 186μs), Choco-Q can achieve a success rate over 50%.

E. Mid-Measurement and Post-Selection

Here, we evaluate the performance of Choco-Q with the mid-measurement and post-selection (MMPS) optimization. Fig. 12 demonstrates the benefit of using mmps under a Pauli noise model with 2×10^{-3} error rate. We can see that MMPS helps to guarantee a near 100% in-constraints rate, even the original in-constraints rate of larger problems like G2, K2 is lower than 25%. This 100% in-constraints rate further leads to a $5.13\times$ improvement on success rate. This improvement is particularly pronounced in benchmarks involving more than 10 qubits. Especially on the G2 benchmark with 15 qubits, the improvement goes to $8.32\times$. For example, the success rate of the K2 problem increases from 1.6% to 12.3%, making the 18-qubit problem solvable in noisy circumstances. The improvements in the success rate come from the amplification of the probability of measuring the valid states satisfying the constraints, using the post-selection technique. Besides, this post-selection is performed layer by layer, which hinders the propagation of unsatisfactory solutions.

VII. Related Work

A. Quantum Algorithm for Constrained Binary Optimization

The first quantum approach to this problem is quantum annealing [41], which uses the quantum adiabatic theorem to solve unconstrained binary optimization. This approach suffers from long embedding time and long evolution time. To overcome this limit, QAOA provides a gate-model version of quantum annealing and uses a variation quantum algorithm to shorten evolution time. Since QAOA can be deployed on current noisy quantum devices, it has been studied and optimized from different aspects, such as parameter initialization [8], parameter updating [42], and distributed version [12]. However, quantum annealing and QAOA cannot deal with constraints.

There are several approaches trying to address the low success rates and long execution times. Zichang et al. [43] find that initializing the QAOA with the ground state of the cyclic driver Hamiltonian can improve the performance, however, significantly limited by hardware noise. Q-CHOP [44] constructs a continuous Hamiltonian path that maintains the system within a feasible solution space. But it fails to theoretically ensure 100% in-constraint rate. Similarly, Franz G. Fuchs et al. [45] extends the cyclic Hamiltonian to preserve different formats of constraints,

which requires prior knowledge of all possible solutions, resulting huge computational cost. This paper gives a universal approach by incorporating the commute Hamiltonian into QAOA and developing an efficient compilation flow, ensuring fast and reliable solutions across various constraints.

Some universal quantum algorithms, alternating from quantum annealing and QAOA, are used to solve this problem with specialized modification. One is the hardware efficient ansatz (HEA) [38]. Each variable is encoded by one qubit, and the objective is modified like penalty-QAOA. Although hardware-efficient, this method cannot always converge into an optimal solution since the circuit structure is not specialized. Another approach named Grover adaptive search uses the Grover algorithm, tailored with a selection circuit to exclude the solution outside the constraints [40]. However, the selection circuit is too complex to deploy on hardware. The search process generates too many solutions outside the constraints, requiring enormous iteration to find the target solution.

VIII. Conclusion

This paper proposes Choco-Q, a fast and hardware-efficient approach for constrained binary optimization problems, which applies commute Hamiltonian that supports arbitrary linear constraints. Then, we develop three optimization techniques to reduce the circuit depth, including Hamiltonian serialization, equivalent decomposition, and variable elimination. These three techniques work together, shrinking the circuit depth from tens of thousands to hundreds. In conclusion, our design greatly improves the success rate and reduces the overall latency of QAOA.

References

[1] R. Z. Farahani and M. Hekmatfar, *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009.

[2] W. Herroelen, "Project scheduling—theory and practice," *Production and Operations Management*, vol. 14, no. 4, 2005.

[3] S. Benati and R. Rizzi, "A mixed integer linear programming formulation of the optimal mean/value-at-risk portfolio problem," *European Journal of Operational Research*, vol. 176, no. 1, 2007.

[4] D. Dugošija, A. Savić, and Z. Maksimović, "A new integer linear programming formulation for the problem of political districting," *Annals of Operations Research*, vol. 288, 2020.

[5] A. Omu, R. Choudhary, and A. Boies, "Distributed energy resource system optimisation using mixed integer linear programming," *Energy Policy*, vol. 61, 2013.

[6] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[7] A. Verma and M. Lewis, "Penalty and partitioning techniques to improve performance of qubo solvers," *Discrete Optimization*, vol. 44, 2022.

[8] 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 (ASPLOS)*, 2024.

[9] T. Yoshioka, K. Sasada, Y. Nakano, and K. Fujii, "Experimental demonstration of fermionic qaoa with one-dimensional cyclic driver hamiltonian," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 1, 2023.

[10] I. Q. Devices, "ibmq quito," 2024. [Online]. Available: https://quantum-computing.ibm.com/services/resources

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

[12] R. Ayanzadeh, N. Alavisamani, P. Das, and M. Qureshi, "Frozenqubits: Boosting fidelity of qaoa by skipping hotspot nodes," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023.

[13] M. Ayodele, "Penalty weights in qubo formulations: Permutation problems," in *European Conference on Evolutionary Computation in Combinatorial Optimization (EvoStar)*, 2022.

[14] S. Hadfield, Z. Wang, E. G. Rieffel, B. O'Gorman, D. Venturelli, and R. Biswas, "Quantum approximate optimization with hard and soft constraints," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing (PMES)*, 2017.

[15] T. R. Jensen and B. Toft, *Graph coloring problems*. John Wiley & Sons, 2011.

[16] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Physical Review E*, vol. 58, no. 5, 1998.

[17] S. Hadfield, Z. Wang, B. O'gorman, E. G. Rieffel, D. Venturelli, and R. Biswas, "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz," *Algorithms*, vol. 12, no. 2, 2019.

[18] W. Heisenberg, *The physical principles of the quantum theory*. Courier Corporation, 1949.

[19] D. Mac Kernan, G. Ciccotti, and R. Kapral, "Trotter-based simulation of quantum-classical dynamics," *The Journal of Physical Chemistry B*, vol. 112, no. 2, 2008.

[20] M. A. de Gosson, "Born–jordan quantization and the equivalence of the schrödinger and heisenberg pictures," *Foundations of Physics*, vol. 44, no. 10, 2014.

[21] H. Leipold and F. M. Spedalieri, "Constructing driver hamiltonians for optimization problems with linear constraints," *Quantum Science and Technology*, vol. 7, no. 1, 2021.

[22] C. Chen, B. Schmitt, H. Zhang, L. S. Bishop, and A. Javadi-Abhar, "Optimizing quantum circuit synthesis for permutations using recursion," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, 2022.

[23] A. Xu, A. Molavi, L. Pick, S. Tannu, and A. Albarghouthi, "Synthesizing quantum-circuit optimizers," *Proceedings of the ACM on Programming Languages (PLDI)*, 2023.

[24] G. H. Low and I. L. Chuang, "Optimal hamiltonian simulation by quantum signal processing," *Physical Review Letters*, vol. 118, no. 1, 2017.

[25] C. Gidney, "Constructing large controlled nots," 2015. [Online]. Available: https://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Nots.html

[26] K. Jain and V. V. Vazirani, "Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation," *Journal of the ACM (JACM)*, vol. 48, no. 2, 2001.

[27] W.-J. van Hoeve, "Graph coloring with decision diagrams," *Mathematical Programming*, vol. 192, no. 1, 2022.

[28] G. Dósa, "The tight bound of first fit decreasing bin-packing algorithm is ffd (i) $\leq$ 11/9 opt (i)+ 6/9," in *International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. Springer, 2007.

[29] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 3, 2013.

[30] P. Slavík, "A tight analysis of the greedy algorithm for set cover," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996.

[31] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Phys. Rev. X*, vol. 10, p. 021067, Jun 2020.

[32] M. T. Melo, S. Nickel, and F. Saldanha-Da-Gama, "Facility location and supply chain management–a review," *European Journal of Operational Research*, vol. 196, no. 2, 2009.

[33] T. N. Bui and B. R. Moon, "Genetic algorithm and graph partitioning," *IEEE Transactions on Computers (TC)*, vol. 45, no. 7, 1996.

[34] Wikipedia contributors, "Identical-machines scheduling — Wikipedia, The Free Encyclopedia," 2024, [Online; accessed 12-October-2024]. [Online]. Available: https://en.wikipedia.org/wiki/Identical-machines_scheduling

[35] A. Caprara, P. Toth, and M. Fischetti, "Algorithms for the set covering problem," *Annals of Operations Research*, vol. 98, no. 1, 2000.

[36] K. N. Bhaskar, "Linear programming and capital budgeting: the financing problem," *Journal of Business Finance and Accounting*, vol. 5, no. 2, 1978.

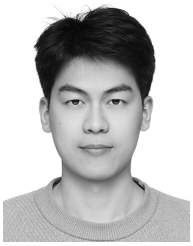
[37] Wikipedia contributors, "Identical-machines scheduling — Wikipedia, The Free Encyclopedia," 2024, [Online; accessed 12-October-2024]. [Online]. Available: https://en.wikipedia.org/wiki/Travelling_salesman_problem

[38] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, 2017.

[39] M. J. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.
[40] A. Gilliam, S. Woerner, and C. Gonciulea, "Grover adaptive search for constrained polynomial binary optimization," *Quantum*, vol. 5, 2021.
[41] G. E. Santoro and E. Tosatti, "Optimization using quantum mechanics: quantum annealing through adiabatic evolution," *Journal of Physics A: Mathematical and General*, vol. 39, no. 36, 2006.
[42] M. Streif and M. Leib, "Training the quantum approximate optimization algorithm without access to a quantum processing unit," *Quantum Science and Technology*, vol. 5, no. 3, 2020.
[43] Z. He, R. Shaydulin, S. Chakrabarti, D. Herman, C. Li, Y. Sun, and M. Pistoia, "Alignment between initial state and mixer improves qaoa performance for constrained optimization," *npj Quantum Information*, vol. 9, no. 1, 2023.
[44] P. Gokhale, "Q-chop: Quantum constrained hamiltonian optimization," in *APS March Meeting Abstracts*, vol. 2022, 2022.
[45] F. G. Fuchs, K. O. Lye, H. Møll Nilsen, A. J. Stasik, and G. Sartor, "Constraint preserving mixers for the quantum approximate optimization algorithm," *Algorithms*, vol. 15, no. 6, 2022.

Debin Xiang received the B.S. degree in Physics from Zhejiang University, in 2024. He is currently working toward a PhD degree with the College of Computer Science, Zhejiang University, Hangzhou. His research interests include quantum computing software and architecture optimization.

Liqiang Lu is an assistant professor (ZJU100 Young Professor) in the College of Computer Science, Zhejiang University (ZJU), China. His research interests include quantum computing, computer architecture, deep learning accelerator, and software-hardware codesign. He has authored more than 30 scientific publications in premier international journals and conferences in related domains, including ISCA, MICRO, HPCA, ASPLOS, FCCM, DAC, IEEE Micro, and TCAD. He also serves as a TPC member in the premier conferences in the related domain, including ICCAD, FPT, HPCC etc.



Qifan Jiang received the B.S. degree in internet of things engineering from Anhui University of Technology. He is currently pursuing an M.S. degree in Software Engineering at Zhejiang University. His research interests include quantum algorithms and their applications.

Siwei Tan received the B.S. degree in 2019 in computer science from Zhejiang University, where he is currently working toward the PhD degree in computer science. His research focuses on the application of interpretable machine learning in social sciences.

Jianwei Yin received the Ph.D. degree in computer science from Zhejiang University (ZJU), in 2001. He was a visiting scholar with the Georgia Institute of Technology. He is currently a full professor with the College of Computer Science, ZJU. Up to now, he has published more than 100 papers in top international journals and conferences. His current research interests include service computing and business process management. He is an associate editor of the IEEE Transactions on Services Computing.