

Rasengan: A Transition Hamiltonian-based Approximation Algorithm for Solving Constrained Binary Optimization Problems

Qifan Jiang
Zhejiang University
Hangzhou, China
jiang7f@zju.edu.cn

Liqiang Lu*
ZJU-Ningbo Global Innovation Center
Zhejiang University
Ningbo, China
liqianglu@zju.edu.cn

Debin Xiang
Zhejiang University
Hangzhou, China
db.xiang@zju.edu.cn

Tianyao Chu
Zhejiang University
Hangzhou, China
tianyao_chu@zju.edu.cn

Tianze Zhu
Zhejiang University
Hangzhou, China
tianzezhu@zju.edu.cn

Jingwen Leng
Shanghai Jiao Tong University
Shanghai, China
leng-jw@cs.sjtu.edu.cn

Yun Liang
Peking University
Beijing, China
ericlyun@pku.edu.cn

Xiaoming Sun
Institute of Computing Technology,
Chinese Academy of Sciences
Beijing, China
sunxiaoming@ict.ac.cn

Jianwei Yin*
ZJU-Ningbo Global Innovation Center
Zhejiang University
Ningbo, China
zjuyjw@cs.zju.edu.cn

Abstract

Constrained binary optimization is a representative NP-hard problem in various domains, including engineering, scheduling, and finance. Variational quantum algorithms (VQAs) provide a promising methodology for solving this problem by integrating the power of quantum parallelism and classical optimizer. However, existing methods fail to achieve both high accuracy and low circuit complexity, rendering it impossible to deploy onto current quantum devices. This paper proposes Rasengan, a high-precision and deployable approach that leverages algorithm-hardware codesign for solving constrained optimization problems. Unlike traditional VQAs that shrink the whole space and locate the possible solutions in a superposition state, our key idea is to expand the search space from one feasible solution and precisely identify the optimal solution in a basis state. Specifically, we propose the transition Hamiltonian that exponentially explores the entire feasible solution space with all combinations of homogeneous basis vectors. We then introduce three optimization techniques, which greatly reduce the circuit complexity when implementing the Hamiltonian simulation, including Hamiltonian simplification and pruning, segmented execution, and solution purification. Experiments demonstrate that Rasengan improves the accuracy by 4.12 \times compared to SOTA QAOA [43] and exhibits 379 \times improvement on real-world quantum platforms.

CCS Concepts

• Computer systems organization \rightarrow Quantum computing.

*Corresponding Author



This work is licensed under a Creative Commons Attribution 4.0 International License.
MICRO '25, Seoul, Republic of Korea
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1573-0/25/10
<https://doi.org/10.1145/3725843.3756107>

Keywords

Variational quantum algorithms, Hamiltonian simulation, Constrained binary optimization

ACM Reference Format:

Qifan Jiang, Liqiang Lu, Debin Xiang, Tianyao Chu, Tianze Zhu, Jingwen Leng, Yun Liang, Xiaoming Sun, and Jianwei Yin. 2025. Rasengan: A Transition Hamiltonian-based Approximation Algorithm for Solving Constrained Binary Optimization Problems. In *58th IEEE/ACM International Symposium on Microarchitecture (MICRO '25)*, October 18–22, 2025, Seoul, Republic of Korea. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3725843.3756107>

1 Introduction

In the combination optimization field, constrained binary optimization problems are to find the optimal value of decision variables that can only take binary values (0 or 1), while subjecting to certain constraints [27]. These problems are extensively applied in fields such as resource allocation [13], route optimization [16], engineering planning [31], and financial investment [5]. Since constrained binary optimization problems are typically NP-hard [2], the time and space complexity to find the exact optimal solution using classical computers increase exponentially with problem size.

Quantum computing, with its unique capabilities such as superposition and entanglement, has demonstrated its advantage for certain computational tasks theoretically [26, 35]. With the rapid advancement of quantum hardware, quantum computing is showing unprecedented potential in addressing combinatorial optimization problems. In particular, as a type of hybrid quantum-classical algorithm, variational quantum algorithms (VQAs) have become outstanding in the NISQ era [38, 44] for their relatively high deployability. They encode binary variables by qubits and create a gate-model circuit to parameterize the distribution of possible solutions [4, 22, 24, 39]. The circuit then can be deployed to quantum devices, like superconducting, ion trap, and neural atom

Table 1: VQA designs for constrained binary optimization.

	Penalty-term-based [39]			Hamiltonian-based	
Methods	HEA [24] (Nature'17)	Frozen-Q[3] (ASPLOS'23)	Red-QAOA[40] (ASPLOS'24)	Choco-Q [43] (HPCA'25)	Rasengan
Feature	hardware efficient	frozen qubit	energy preservation	commute Hamilt.	transition Hamilt.
Output state	superposition state involved with poor solutions				basis state
ARG (↓)	1100	~1000		7.27	0.70
Latency	702 ms	~300 ms		445 ms	144 ms

* The latency includes the classical and quantum parts, without data communication time. The execution time is based on the IBM Quesec model [11].

platforms. The classical includes an optimizer to iteratively tune the parameters of the circuit until converging to the optimal objective.

The VQAs for constrained binary optimization can be categorized into hardware-efficient ansatz (HEA) [24], and quantum approximate optimization algorithm (QAOA) [15, 41, 44]. The circuit of HEA contains repeated layers of native rotation gates and entangling gates. This adaptive structure makes it hardware-efficient in NISQ devices. QAOA framework alternatively applies the evolution unitary of an objective Hamiltonian and a mixer Hamiltonian, adjusting the evolution time of each Hamiltonian to search for the optimal solution. In particular, penalty-term-based QAOA (P-QAOA) [3, 4, 39, 40] inserts the constraints into the objective Hamiltonian as penalty terms. While, commute-Hamiltonian-based QAOA (Choco-Q) [22, 29, 43] embeds the constraints by designing the mixer Hamiltonian to commute with constraints.

Table 1 summarizes the features and metrics of prior VQA designs, tested on the set covering problem [8] using a 12-qubit noise-free simulator. Here, we use the approximation ratio gap (ARG), which quantifies solution quality as the gap between the algorithm's output and the optimum, with lower values indicating better performance. We observe that previous methods inevitably encounter the dilemma between accuracy and latency. Specifically, penalty-term-based approaches exhibit inferior performance, with around one thousand ARG, making it essentially impossible to reach the optimum. Even with advanced QAOA optimization techniques, such as FrozenQubits [3] to reduce circuit depth and Red-QAOA [40] for optimizing initial parameters, the penalty-based method remains ineffective due to its limited capability in enforcing constraints. On the other hand, though existing Hamiltonian-based QAOA like Choco-Q [43] achieve a better ARG, they incur higher costs in encoding specialized mixer Hamiltonians, leading to greater circuit depth and latency.

Both QAOA and HEA face inherent expressivity limits and cannot reliably converge to a single target basis state. They achieve the feasible solution space by iteratively shrinking the search space, leaving the final state as a superposition of good and poor solutions, which degrades quality. QAOA is based on the discretized form of adiabatic quantum computation [15]. Theoretically, it requires an infinitely deep circuit to fully realize adiabatic evolution so that it can shrink to the feasible solution space. HEA creates a superposition state using parameterized rotation gates, followed by entangling gates to determine the direction of search space restriction. To enhance expressibility, HEA necessitates multiple layers of

repetition. However, since the feasible solution space occupies only a small fraction of the entire quantum state space (e.g., 72/4096 in the case used in Table 1), this shrinking process demands significant circuit depth to maintain accuracy.

In this paper, we propose Rasengan, a high-precision and low-complexity algorithm for solving the constrained binary optimization problem. Understanding that traditional VQAs rely on global superposition to exhaustively explore and shrink the search space with tremendous non-feasible solutions, we inversely design the algorithm that expands the search space from one arbitrary feasible solution. In linear algebra, a single feasible solution can span the full solution space via homogeneous basis vectors. Building on this insight, we propose the *transition Hamiltonian* to exponentially construct the accurate space with feasible solutions. Such expansion is able to cover all feasible solutions and keep the space only containing the feasible solutions. Different from prior algorithms that output a superposition state entangled with poor solutions, Rasengan has a much higher probability of identifying the optimal solution in a basis state.

Then, we propose three algorithm-hardware codesign optimizations to improve the deployment on current noisy quantum hardware. First, we reconstruct the homogeneous basis to simplify the transition Hamiltonians and prune redundant those that contribute to no expansion of the feasible solution space. Second, we introduce a segmented execution strategy that partitions the sequence of transition Hamiltonians into segments and executes the segments individually. Finally, we employ an error mitigation technique by purifying the noisy solutions after each segment and removing the infeasible ones.

Overall, the contributions of this work are as follows:

- We propose a novel and deployable variational quantum algorithm to solve constrained binary optimization. Leveraging transition Hamiltonian, our design can cover all feasible solutions (noise-free) with low circuit complexity.
- We propose a series of optimization techniques to improve the deployability and accuracy in noisy hardware. The circuit depth is reduced from ~7000 to ~50, making Rasengan deployable on modern quantum platforms.
- We conduct rigorous experiments on simulators and real-world quantum devices. Rasengan shows notable improvement in both accuracy and latency compared to prior works.

At the algorithmic level, across 2000 cases from five domains, Rasengan increases the accuracy by 4.12× and reduces the circuit depth by 1.96× compared to the state-of-the-art prior works. On real-world quantum hardware, Rasengan is the first quantum algorithm to beat the mean quality of feasible solution baseline in our test, achieving a 379× improvement. Besides, our Hamiltonian simplification and pruning, and probability-preserving segmented execution techniques reduce the circuit depth by over 94.6%. Our error mitigation technique improves the accuracy by more than 303×. Rasengan is publicly available on <https://github.com/JanusQ/rasengan>.

2 Background

2.1 Constrained Binary Optimization

The constrained binary optimization problem optimizes an objective function while satisfying specified constraints. The objective

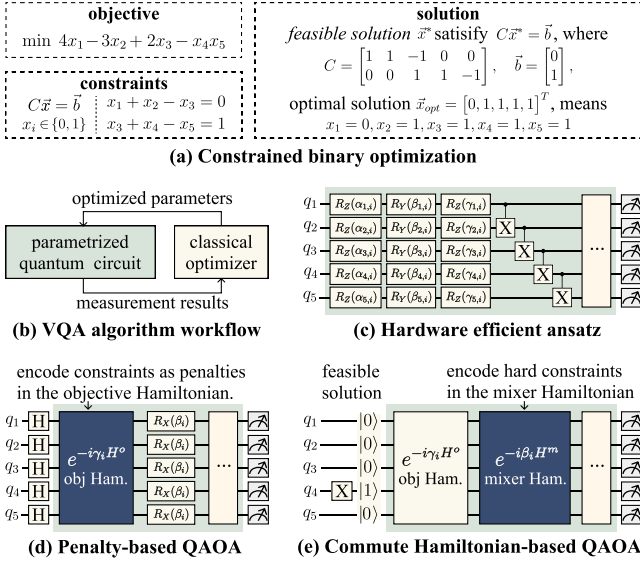


Figure 1: The constrained binary optimization problems and the variational quantum algorithms for solving it.

function takes binary variables as inputs and returns a scalar. The constraints are expressed as linear equalities or inequalities. The inequality constraints can be transformed into equality using auxiliary binary variables. Mathematically, the problem can be formulated as follows:

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

Where C is the matrix representing the coefficients of the equality constraints, and \vec{b} is a constant vector representing the constraint bounds. In the constrained optimization, if \vec{x}^* satisfies the constraints, i.e., $C\vec{x}^* = \vec{b}$, then \vec{x}^* is called a *feasible solution*. Figure 1 (a) gives an example with five variables and two constraints.

By softly embedding the constraints through penalty terms, the problem can be transformed into an unconstrained binary optimization form:

$$\min_{\vec{x}} \text{ or } \max_{\vec{x}} \quad f(\vec{x}) + \lambda \|C\vec{x} - \vec{b}\|$$

where λ is the penalty coefficient that indicates the strength of the penalty.

2.2 Variational Quantum Algorithms

Variational quantum algorithms (VQAs) are a class of hybrid quantum-classical algorithms [32]. As shown in Figure 1 (b), VQAs use parameterized quantum circuits to harness quantum parallelism, and update parameters via a classical optimizer to reach the optimal objective. There are several VQAs for solving constrained binary optimization problems, including hardware efficient ansatz (HEA) [24] and the quantum approximate optimization algorithms (QAOA) [44]. Figure 1 (c) gives the quantum circuit of HEA, which is designed for NISQ devices. It contains repeated layers of native local rotations like Rotation-Z (R_Z), and Rotation-Y (R_Y) gates and entangling operations like Control-X (CX) gate. These gate operations are organized according to the hardware topology, making HEA

suitable for current NISQ devices. QAOA takes the discretized form of adiabatic quantum computing for quantum circuit construction. It alternates the time evolution of the objective Hamiltonian and the mixer Hamiltonian. The objective Hamiltonian encodes the objective function and the mixer Hamiltonian explores the solution space. QAOA then adjusts the evolution time parameters by the classical optimizer to approach the optimal solution.

Native QAOA, where its mixer Hamiltonian is a layer of Rotation-X (R_X) gates, can only handle unconstrained problems. For constrained problems, there are mainly two extensions. One is the penalty-term-based QAOA [4, 39], where the constraints are encoded by incorporating penalty terms into the objective Hamiltonian (H^o), as shown in Figure 1 (d). Another is the commute-Hamiltonian-based QAOA [22, 29], which designed a new mixer Hamiltonian (H^m) to encode the constraints, as shown in Figure 1 (e). The new mixer Hamiltonian must commute with the constraints operators, and the initial state needs to be prepared as one feasible solution. By the two conditions, commute-Hamiltonian-based QAOA ensures that all output quantum states satisfy the constraints.

At the end of the circuit in VQA, we perform measurements to obtain the basis state, which can also be represented by a binary vector \vec{v} . If \vec{v} is a feasible solution, then we can also call the corresponding basis state a feasible state.

2.3 Hamiltonian Simulation

The Hamiltonian in the QAOA circuit is implemented by Hamiltonian simulation that uses quantum computers to simulate the time evolution of a physical system under a Hamiltonian. Fundamentally, the Hamiltonian describes the energy landscape of the system, which controls the quantum state evolution by Schrödinger's equation:

$$|\psi(t)\rangle = e^{-iHt/\hbar} |\psi(0)\rangle,$$

where $|\psi(t)\rangle$ is the quantum state at time t , H is the Hamiltonian, \hbar is the reduced Planck constant, and $|\psi(0)\rangle$ is the initial state. We can see that the time evolution operator $e^{-iHt/\hbar}$ is responsible for the transformation between two quantum states. In particular, by applying the Taylor expansion to the time evolution operator, one can derive that if the Hamiltonian H satisfies $H^2 |\psi(0)\rangle = |\psi(0)\rangle$, the time evolution simplifies to:

$$|\psi(t)\rangle = \cos\left(\frac{t}{\hbar}\right) I |\psi(0)\rangle - i \sin\left(\frac{t}{\hbar}\right) H |\psi(0)\rangle. \quad (2)$$

This equation indicates that the quantum state after time evolution is a superposition of two parts: the first part $I |\psi(0)\rangle$ maintains the initial state, while the second part $H |\psi(0)\rangle$ applies the Hamiltonian H to the initial state. The amplitudes of the two parts are determined by the cosine and sine functions, respectively, related to the time parameter t .

3 Algorithm Design

Existing variational quantum algorithms are limited by either the deployability on noisy devices or the poor accuracy due to insufficient searching. For example, HEA facilitates hardware implementation by sacrificing the expressivity in the solution space, making it difficult to effectively capture the target quantum state. In contrast,

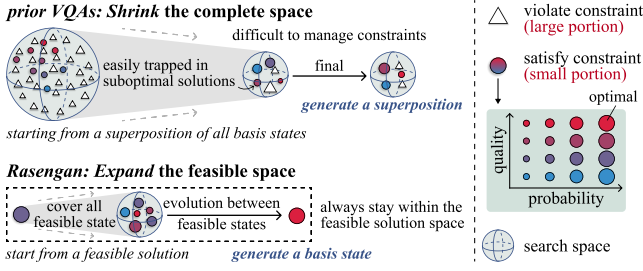


Figure 2: Methodology comparison between prior VQAs and Rasengan, regarding solution space exploration.

QAOA features a discretized form of adiabatic quantum computation and alternates between the objective Hamiltonian and the mixer Hamiltonian, requiring exhaustively searching the solution space until reaching high accuracy.

Fundamentally, both HEA and QAOA try to shrink the search space, starting from a superposition of all basis states and gradually locating the feasible solution space, as shown in Figure 2. Furthermore, since the feasible solution space constitutes only a small fraction of the total search space, the shrinking scheme is inefficient and prone to trapping in suboptimal solutions (see large purple-red points in Figure 2). *Ideally, the quantum algorithm should collapse the quantum state to the optimal solution.* However, as the space of HEA and QAOA is much larger than the feasible solution space and exhibits low expressivity, the final state often turns out to be a superposition with poor solutions, degrading output quality.

To perfectly form the feasible solution space, we design the quantum algorithm inversely. Instead of shrinking the space, our key insight lies in the expansion of the space growing out of one feasible solution. Clearly, we need to ensure that 1) each expansion should keep the space only containing the feasible solutions, 2) the final space after expansion should cover all feasible solutions so that we can find the optimal ones. To this end, we formulate the expansion-based quantum algorithm using the general solution theory of linear algebra.

As shown in Figure 3 (a), for a system with linear constraints $C\vec{x} = \vec{b}$, one feasible solution can be derived from another feasible solution by accumulating with the linear combination of homogeneous basis $\{\vec{u}\}$ [21], where \vec{u} is the solution of equation $C\vec{u} = \vec{0}$. In other words, given a feasible solution \vec{x}_p that meets $C\vec{x}_p = \vec{b}$, any feasible solution \vec{x}_g can be calculated as:

$$\vec{x}_g = \vec{x}_p + \vec{u}, \quad \forall C\vec{u} = \vec{0} \quad (3)$$

For example, consider the case shown in Figure 1 (a), in which

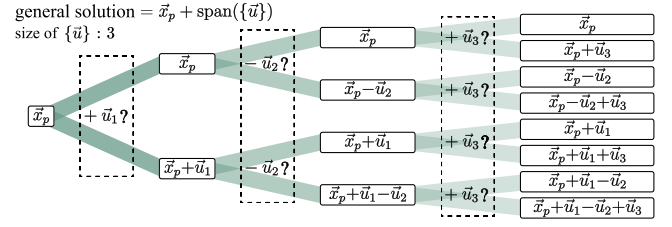
$$C = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \vec{x}_p = [0, 0, 0, 1, 0]^T.$$

Solving $C\vec{u} = \vec{0}$ yields the homogeneous basis:

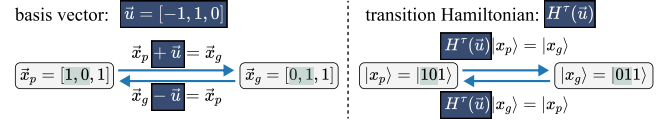
$$\vec{u}_1 = [-1, 1, 0, 0, 0]^T, \vec{u}_2 = [-1, 0, -1, 1, 0]^T, \vec{u}_3 = [1, 0, 1, 0, 1]^T \quad (4)$$

Thus, by leveraging the span of the homogeneous basis and kicking out non-binary strings (e.g., $[-1, 0, -1, 2, 0]$), we can obtain all feasible solutions:

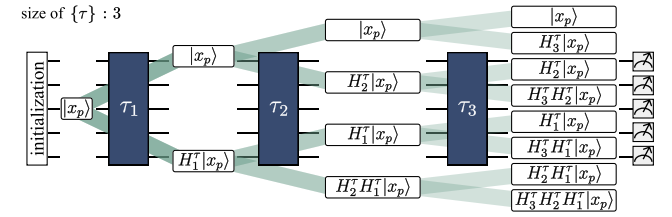
$$\begin{aligned} \vec{x}_2 &= \vec{x}_p - \vec{u}_2 = [1, 0, 1, 0, 0]^T, & \vec{x}_4 &= \vec{x}_p - \vec{u}_2 + \vec{u}_1 = [0, 1, 1, 0, 0]^T, \\ \vec{x}_3 &= \vec{x}_p + \vec{u}_3 = [1, 0, 1, 1, 1]^T, & \dots \end{aligned}$$



(a) The linear combination of basis vectors spans the entire solution space.



(b) Transition Hamiltonian for homogeneous basis operations.



(c) The transition Hamiltonian simulations exponentially expands the entire solution space.

Figure 3: Applying Transition Hamiltonian to precisely explore the entire space of feasible solutions.

That is to say, the homogeneous basis $\{\vec{u}\}$ can generate the entire feasible solution space with one feasible solution \vec{x}_p calculated in advance. Building upon this, we introduce a transition mechanism that effectively navigates the feasible solution space by leveraging the properties of the homogeneous basis.

3.1 Transition Hamiltonian Formulation

The expansion mechanism provided by the homogeneous basis suggests a natural way to explore the feasible solution space, leveraging quantum parallelism. The classical version relies on explicit linear combinations, while quantum computing offers an alternative through Hamiltonian simulation, which can create superposition states and exponentially accelerate space expansion. The key challenge is: 1) ensuring that the quantum state evolution remains confined within the feasible solution space, as directly applying Hamiltonian simulation could lead to wrong solutions that violate problem constraints; 2) the search space should record every feasible solution during the expansion, e.g., both \vec{x}_p and \vec{x}_g should be kept.

To address challenge 1), we utilize the inherent property that adding or subtracting a homogeneous basis vector to a feasible solution always yields another feasible solution. In the left part of Figure 3 (b), a feasible solution \vec{x}_p produces another feasible solution \vec{x}_g by combining with a homogeneous vector $\vec{u} = \vec{x}_g - \vec{x}_p$. Following Equation 3, we propose the transition Hamiltonian that operates just like the homogeneous basis expansion. Concretely, the transition Hamiltonian is designed to evolve one feasible basis state into another while meeting the linear constraints. In the right part of Figure 3 (b), a feasible solution $|x_p\rangle$ under the operator

of $H^\tau(\vec{u})$ evolves to another feasible solution $|x_g\rangle$. More generally, given two feasible quantum basis states $|x_p\rangle$ and $|x_g\rangle$, the transition Hamiltonian $H^\tau(\vec{u})$ enables their mutual conversion:

$$H^\tau(\vec{u}) |x_p\rangle = |x_g\rangle, H^\tau(\vec{u}) |x_g\rangle = |x_p\rangle \quad (5)$$

For challenge 2), according to Equation (5), we have $H^\tau(\vec{u})^2 |x_p\rangle = |x_p\rangle$. Together with the Hamiltonian simulation in Equation (2), the output state after applying transition Hamiltonian to $|x_p\rangle$ is:

$$e^{-iH^\tau(\vec{u})t} |x_p\rangle = \cos(t) |x_p\rangle - i \sin(t) |x_g\rangle \quad (6)$$

We can see that both $|x_p\rangle$ and $|x_g\rangle$ are kept after conducting the Hamiltonian simulation $e^{-iH^\tau(\vec{u})t}$ on $|x_p\rangle$. In other words, the Hamiltonian simulation of the transition Hamiltonian enlarges the feasible solution space through superposition. As shown in Figure 3 (c), instead of enumerating all $2^3 = 8$ linear combinations of three homogeneous basis vectors individually, we only need to perform three simulations of transition Hamiltonians to generate a superposition of all feasible solutions, thereby achieving an exponential expansion of the solution space. Next, we give the formal definition of the transition Hamiltonian.

DEFINITION 1. Transition Hamiltonian. Considering two feasible basis states $|x_p\rangle$ and $|x_g\rangle$, with their distance equals to the homogeneous basis vectors $\vec{u} = \vec{x}_g - \vec{x}_p$, the transition Hamiltonian is defined as:

$$H^\tau(\vec{u}) = \bigotimes_{i=1}^n \sigma^+(u^i) + \bigotimes_{i=1}^n \sigma^-(u^i) \quad (7)$$

$$\vec{u} = \begin{bmatrix} u^1 \\ \vdots \\ u^n \end{bmatrix}, \quad \sigma(u^i) = \begin{cases} \sigma^+, & u^i = 1 \\ \sigma^-, & u^i = -1 \\ I, & u^i = 0 \end{cases}$$

where σ^+ and σ^- are the raising and lowering operators [12].

The number of transition Hamiltonians is equal to the number of homogeneous basis vectors. We then demonstrate that finite transition simulations can cover all feasible solutions that satisfy the constraints.

THEOREM 1. We assume that there are m transition Hamiltonians derived from the linear constraints. For totally unimodular matrices, given one feasible solution, repeating Hamiltonian simulations of these m transition Hamiltonians for m times (m^2 in total) can cover all feasible solutions. For more complex cases, the upper bound is m^3 .

PROOF. Any feasible solution \vec{x}_g can be expressed as the linear combination of homogeneous basis $\{\vec{u}\}$ with one pre-calculated feasible solution \vec{x}_p .

$$\vec{x}_g = \vec{x}_p + \sum_{k=1}^m a_k \vec{u}_k \quad (8)$$

where $a_k \in \{-1, 0, 1\}$ and $\vec{u}_k \in \{-1, 0, 1\}^{\otimes n}$ since the homogeneous basis vectors are linearly independent and the constraints of \vec{x}_p and \vec{x}_g are binaries. Thus, the entire feasible space can be generated by permuting the subsets of the homogeneous basis $\{\vec{u}\}$.

According to Equation (7), we can demonstrate that, for the quantum state $|\varphi_\perp\rangle$ that turns to be a non-binary state after its vector adding or subtracting \vec{u} , the transition Hamiltonian will

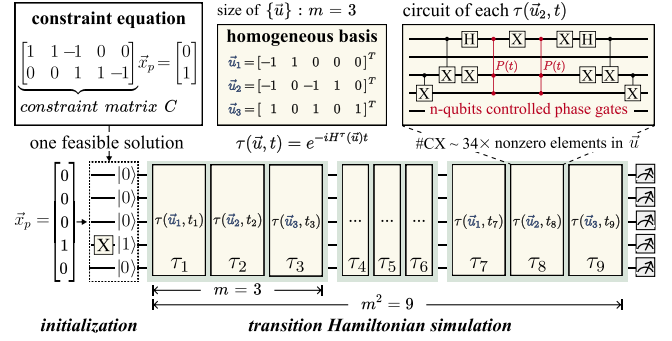


Figure 4: Circuit implementation for transition Hamiltonian. This example is consistent with Figure 1 (a) and Equation (4).

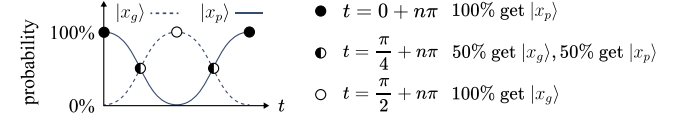
change $|\varphi_\perp\rangle$ to 0. In other words, by expanding the Taylor series, the transition Hamiltonian simulation keeps $|\varphi_\perp\rangle$ the same:

$$H^\tau(\vec{u}) |\varphi_\perp\rangle = 0$$

$$\Rightarrow e^{-iH^\tau(\vec{u})t} |\varphi_\perp\rangle = |\varphi_\perp\rangle$$

During evolution, the transition Hamiltonian generates new feasible solutions by adding or subtracting homogeneous basis vectors. For totally unimodular matrices, enumerating all permutations of transition Hamiltonians suffices to cover the entire feasible space, which can be achieved by repeating the m basis vectors for m rounds. For more complex matrices, a single enumeration yields one effective application of a basis vector, and m such enumerations are sufficient to cover all feasible solutions. \square

Theorem 1 demonstrates the high expressivity that ensures the coverage of the entire feasible space via transition Hamiltonians. According to Equation (6), the amplitude of each feasible solution is parameterized by the evolution time.



This means, by setting simulation times t , the quantum state can be a basis state of $|x_g\rangle$ or $|x_p\rangle$. Considering that the final optimal solution is also a basis state, theoretically, we can tune the simulation time to locate the optimal feasible solution. Unlike prior methods that always keep in the superposition state that may involve poor solutions, our algorithm shows a higher probability of evolving to the optimal solution.

3.2 Circuit Implementation

Figure 4 illustrates the circuit design to implement the transition Hamiltonian simulation. The initialization is set by one arbitrary feasible solution. Then, the Hamiltonian simulation of m transition Hamiltonians $H^\tau(\vec{u})$, defined by Equation (7), are sequentially repeated for m times, resulting in m^2 transition operators $\tau(\vec{u}, t) = e^{-iH^\tau(\vec{u})t}$.

We prove that the transition operator $\tau(\vec{u}, t)$ can be equally decomposed as a symmetrical circuit structure with two multi-controlled phase gates, as shown in Figure 4. Mathematically, this decomposition ensures the linear complexity that contains $34k$ CX gates, where k is the number of nonzero elements in \vec{u} [20]. As

Algorithm 1: Hamiltonian simplification

Input : Homogeneous basis $U = \{\vec{u}_1, \vec{u}_2, \dots, \vec{u}_m\}$, where $\vec{u}_i \in \{-1, 0, 1\}^{\otimes n}$.

Output : Reconstructed homogeneous basis U' with fewer nonzero elements.

```

1 isValid( $\vec{u}$ ) : Checks if vector  $\vec{u} \in \{-1, 0, 1\}^{\otimes n}$ .
2 nonZero( $\vec{u}$ ) : the number of nonzero elements in  $\vec{u}$ .
3  $U' \leftarrow U$ 
4 for  $i \leftarrow 1$  to  $|U'|$  do
5   for  $j \leftarrow i + 1$  to  $|U'|$  do
6      $\vec{u}_{\text{add}} \leftarrow \vec{u}_i + \vec{u}_j, \vec{u}_{\text{sub}} \leftarrow \vec{u}_i - \vec{u}_j$ 
7     if isValid( $\vec{u}_{\text{add}}$ ) and nonZero( $\vec{u}_{\text{add}}$ ) < nonZero( $\vec{u}_i$ ) then
8        $\vec{u}_i \leftarrow \vec{u}_{\text{add}}$ 
9     if isValid( $\vec{u}_{\text{sub}}$ ) and nonZero( $\vec{u}_{\text{sub}}$ ) < nonZero( $\vec{u}_i$ ) then
10       $\vec{u}_i \leftarrow \vec{u}_{\text{sub}}$ 
11 return  $U'$ 

```

k is always less than or equal to the number of variables n , the upper-bound number of CX gates is $34nm^2$. Such a polynomial cost in circuit depth makes the transition Hamiltonian suitable for hardware deployment. More importantly, compared to the QAOA, which incurs a high cost encoding the objective Hamiltonian for higher-order objective problems, the transition Hamiltonian framework eliminates the need for such encoding, offering greater generality.

4 Circuit Optimization

Although the transition Hamiltonian shows better deployability and generality, the quadratic circuit depth still faces challenges on NISQ devices. For example, solving a graph coloring problem with 24 variables results in ~ 7000 circuit depth, greatly exceeding real-world quantum platforms' executable depth (~ 100). To squeeze the circuit depth, we first try to reduce the number of non-zeros in \vec{u} and prune the redundant transition Hamiltonian. In Section 4.1, we propose to replace the original homogeneous basis vectors with their linear combinations, guided by a greedy algorithm. And the transition Hamiltonian that leads to no expansion of the feasible solution space is pruned. After this, the circuit has around 1000 depth. Furthermore, in Section 4.2, we adopt a segmented execution strategy that groups several transition Hamiltonians into a segment and executes the segment individually. Following this, the circuit depth is reduced to ~ 50 , which is deployable on current quantum devices. Finally, we employ an error mitigation technique to improve the solution quality, which purifies the noisy outputs from each segment (Section 4.3).

4.1 Hamiltonian Simplification and Pruning

Hamiltonian simplification. According to Section 3.2, the complexity of decomposed transition Hamiltonian simulation is linear with the number of nonzero elements in the homogeneous basis vectors. Arithmetically, the number of total nonzero elements may be reduced by choosing another set of basis vectors, calculated from the linear combinations of the original ones. And the transformation of basis vectors still exposes the entire feasible solution space. To simplify the Hamiltonian decomposition, we introduce a greedy-based algorithm to identify a suitable basis that brings fewer nonzero elements, as shown in Algorithm 1.

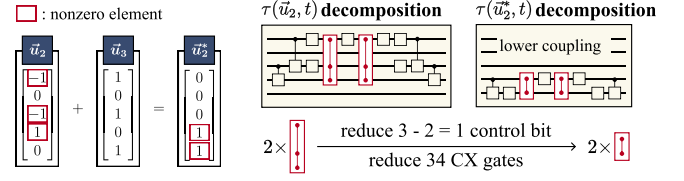
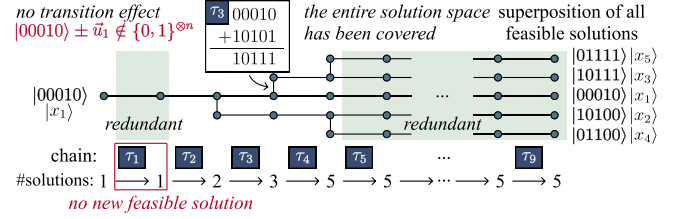
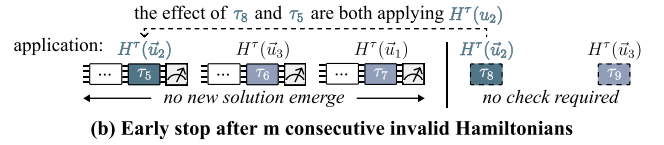


Figure 5: Hamiltonian simplification by reducing the number of nonzero elements in the homogeneous basis vectors.



(a) Pruning the Hamiltonian that introduces no new feasible solution



(b) Early stop after m consecutive invalid Hamiltonians

Figure 6: Pruning redundant transition Hamiltonian that leads to no space expansion of feasible solution.

Specifically, the algorithm iterates over all pairs of homogeneous basis vectors (lines 4-5), exhibiting a time and space complexity of $O(m^2)$. For each pair (\vec{u}_i, \vec{u}_j) , two candidate vectors, \vec{u}_{add} and \vec{u}_{sub} , are generated by addition and subtraction operation (line 6). We then discard the resulting vectors that fall outside the valid homogeneous basis space $\{-1, 0, 1\}^{\otimes n}$, and select the vector with the fewest nonzero elements to replace \vec{u}_i . Since each vector regeneration step has a complexity of $O(n)$, the overall complexity of Algorithm 1 is $O(m^2n)$. Figure 5 provides an example of the Hamiltonian in Figure 4. By adding the basis vector $\vec{u}_3 = [1, 0, 1, 0, 1]$ to the basis vector $\vec{u}_2 = [-1, 0, -1, 1, 0]$, it generates a new vector $\vec{u}_2^* = [0, 0, 0, 1, 1]$, which only has two nonzero elements. Since the original basis vector \vec{u}_2 has three nonzero elements, we replace \vec{u}_2 with \vec{u}_2^* . Under the new basis vectors, two 3-controlled phase gates are simplified to two 2-controlled phase gates, thereby reducing 34 CX gates.

Hamiltonian pruning. Theorem 1 guarantees that m^2 transition Hamiltonians simulations are able to cover all feasible solutions. We observe that some of them are redundant, contributing to no expansion of the feasible solution space, and can be skipped. Taking Figure 6 (a) as an example, there are nine transition Hamiltonians in total with $m = 3$. The first transition Hamiltonian simulation τ_1 does not bring any new feasible solution, making it a redundant Hamiltonian. And after the simulation of the next three transition Hamiltonians (τ_2, τ_3, τ_4), all five feasible solutions have been fully explored, leaving the subsequent Hamiltonians unnecessary.

By executing the circuit m^2 times, we measure the intermediate output and locate the redundant Hamiltonians, where we remove the Hamiltonian if its simulation does not yield a new basis state. Actually, we can eliminate the tail once we detect that m consecutive

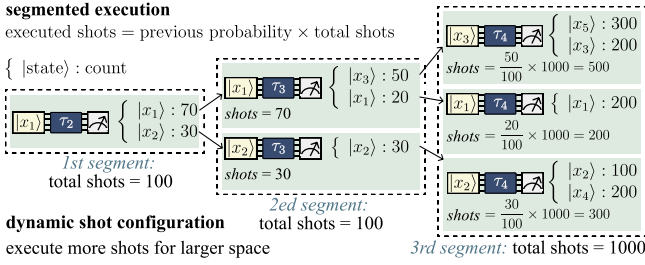


Figure 7: Segmented execution. The state with a higher probability takes more execution shots in the following segment.

Hamiltonian simulations fail to produce new basis states, namely early stop. For example, in Figure 6 (b), we find that the sequence $[\tau_5, \tau_6, \tau_7]$ cannot generate new feasible solutions, the rest Hamiltonian simulation $[\tau_8, \tau_9]$ will be stopped. Note that the complexity of detecting the redundant Hamiltonian is $O(m^2)$. But this is a one-shot process that we can get the redundancy off-line and use this information during VQA training.

4.2 Segmented Execution

After the Hamiltonian simplification and pruning, the circuit depth is reduced from $\sim 10^4$ to $\sim 10^3$, which is still beyond the capability of modern NISQ devices. For instance, solving a graph coloring problem with 24 variables gives rise to a 1008-depth circuit. In the design of variational quantum algorithms, the optimizer updates the parameter guided by the probability distribution from the previous iteration [9]. Unlike traditional QAOAs that apply Hamiltonian simulation to discretely approximate adiabatic evolution, our transition Hamiltonian expands the feasible solution space and distributes the probability to these feasible solutions. Therefore, it exposes the opportunity to partition the transition Hamiltonian sequence and individually execute them by preserving the probability information in each part.

Based on this property, we partition transition Hamiltonians into multiple segments, with each segment having less circuit depth. The output probability from one segment is sent to the next segment as input, where the state with higher probability takes more execution shots in the following segment. For example, in Figure 7, the output of the first segment consists of $|x_1\rangle$ and $|x_2\rangle$, with 70% and 30% measurement probability, respectively. Thus, the execution of the second segment takes 70 shots with $|x_1\rangle$ as input, and 30 shots with $|x_2\rangle$ as input. Besides, the number of shots for each segment can be dynamically configured to balance the trade-off between execution overhead and solution quality. For example, in Figure 7, the number of shots for the third segment is increased by a factor of 10, which exhibits higher precision to preserve the probability information. This trade-off is validated experimentally, favoring more transition Hamiltonians per segment within the decoherence time.

By leveraging segmented execution, each minimal execution circuit depth corresponds to a single transition Hamiltonian simulation. This decomposition reduces the two-qubit gate depth from $34nm^2$, as described in Section 3.2, to $34n$. To achieve an equivalent optimization effect as circuit merging, it suffices to apply a column of X gates to transform the initial state of the current segment into the feasible solution generated by the previous segment. The short

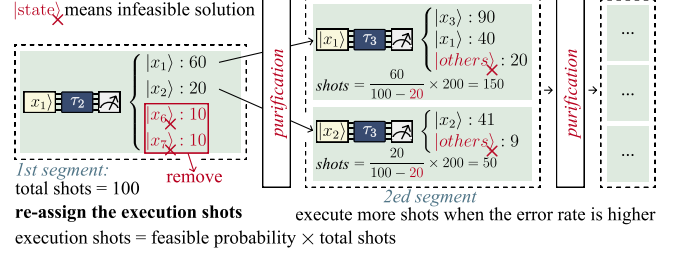


Figure 8: Error mitigation by purifying the noisy solution that does not meet the constraints $C\vec{x} = \vec{b}$.

execution depth and low partitioning overhead make Rasengan's synthesized circuits more manageable.

4.3 Error Mitigation by Purification

Except for compressing circuit depth, the error mitigation technique is an orthogonal optimization for improving deployability. In terms of the transition Hamiltonian, the noise error may navigate the simulation to explore incorrect solutions, gradually destroying the feasibility of the entire solution space. To tackle this, we introduce a purification layer between segments, which validates the solutions after each segment and removes the wrong ones. As shown in Figure 8, we detect that $C\vec{x}_6 \neq \vec{b}$ and $C\vec{x}_7 \neq \vec{b}$. Subsequently, the purified probability distribution is used to determine the number of shots for executing the next segment. For example, when the second segment is configured with 200 shots, the number of shots, with $|x_1\rangle$ as input, is $\frac{60}{100-20} \times 200 = 150$.

The purification is conducted in each iteration by checking the equation $C\vec{x} \neq \vec{b}$ after every segment. However, this checking introduces little overhead compared to the overall VQA training process as it is a matrix-vector multiplication that can be easily accelerated on CPUs and GPUs. For a graph coloring problem with 24 variables, the classical part of training time per iteration is around 700 ms, while the purification takes only 0.05 ms on AMD CPU, which only accounts for less than 0.01% of the total training time.

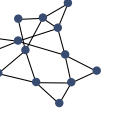
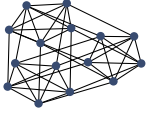
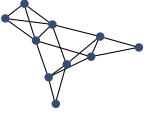
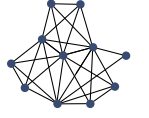
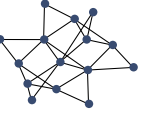
5 Evaluation

5.1 Experiment Setup

Benchmark. We evaluate Rasengan using five practical application scenarios: the facility location problem (FLP) [14], k -partition problem (KPP) [6], job scheduling problem (JSP) [42], set cover problem (SCP) [8], and graph coloring problem (GCP) [23]. For each benchmark, we compile 400 cases from relevant literature [6, 8, 14, 23, 42]. As an example, we use F1 to denote the first scale benchmark in facility location problems. Table 2 summarizes the corresponding number of variables (i.e., qubits) and constraints. To further assess the complexity of the constraints, we analyze their topological structure and present a visualization in the same table. In addition, we compute the average node degree in the constraint topology graph as a measure of constraint hardness.

Complexity of finding a feasible solution. For constrained binary optimization problems, a feasible solution can often be constructed in linear time. For the facility location problem, opening a single facility and assigning all demands to it yields a feasible

Table 2: Algorithmic evaluation on ARG, circuit depth, and the number of parameters under 20 benchmarks.

Benchmark	Facility location [14]				K-partition [6]				Job scheduling [42]				Set covering [8]				Graph coloring [23]					
	F1	F2	F3	F4	K1	K2	K3	K4	J1	J2	J3	J4	S1	S2	S3	S4	G1	G2	G3	G4		
#Variables (#Qubits)	6	15	21	28	8	15	18	21	7	10	14	18	9	12	16	20	12	15	20	24		
#Constraints	3	8	12	15	6	8	9	10	4	5	6	7	4	5	6	7	6	9	8	12		
Initilization complexity	$O(d)$, d : #demands				$O(e)$, e : #elements				$O(j)$, j : #jobs				$O(s)$, s : #sets				$O(g)$, g : #graphs				Improv.	
Graph topology of constraints																						
Average degree	2.42	5.10	6.71	8.68	3.75	6.50	7.75	9.00	3.07	4.05	5.78	7.05	4.55	5.63	7.22	8.69	4.25	5.30	6.55	7.75		
#Feasible solutions	4	24	54	224	4	5	6	7	4	5	13	96	15	72	240	856	9	3	16	64		
ARG	HEA	47.0	157	162	274	354	416	621	1120	65.8	194	193	403	943	1100	1423	1615	177	302	521		796
	P-QAOA	51.4	119	119	194	346	371	493	1184	74.0	161	169	323	836	958	1256	1512	132	226	343	611	1897×
	Choco-Q	0.10	0.50	0.36	0.57	0.32	0.95	1.39	3.50	0.06	0.42	0.54	0.74	3.94	7.27	22.1	26.2	0.22	0.61	1.10	1.53	4.12×
	Rasengan	0.01	0.12	0.14	0.53	0.07	0.89	1.21	3.16	0.01	0.28	0.35	0.55	0.24	0.70	4.02	16.5	0.15	0.43	0.46	0.93	-
Circuit Depth	HEA	46	91	121	156	56	91	106	121	51	66	86	106	61	76	96	116	76	91	116	136	1.96×
	P-QAOA	87	148	174	207	175	250	285	324	110	136	170	194	220	283	390	481	259	310	745	791	6.58×
	Choco-Q	507	1888	2688	3848	1324	3533	4396	5275	1082	2039	3137	4562	1079	1346	1969	2535	1879	1697	3207	3515	49.0×
	Rasengan	34	49	58	58	85	87	87	87	56	69	66	72	29	27	24	22	54	64	40	56	-
#Param.	HEA	90	225	315	420	120	225	270	315	105	150	210	270	135	180	240	300	180	225	300	360	15.7×
	P-QAOA	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0.94×
	Choco-Q	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0.94×
	Rasengan	3	26	53	65	4	8	33	38	4	7	22	28	11	22	49	100	8	6	26	18	-

¹ P-QAOA: penalty-term-based QAOA [39]. Choco-Q: commute-Hamiltonian-based QAOA [43].² We choose the second scale of each problem (F2~G2) to depict the graph topology and the degree is defined as the number of edges connected to the node.

solution in $O(d)$, where d is the number of demands. For the k -partition problem, elements can be greedily assigned into boxes based on capacity, resulting in a feasible solution in $O(e)$, where e is the number of elements. For the job scheduling problem, jobs can be greedily allocated to machines based on their capacity in $O(j)$, where j is the number of jobs. For the set cover problem, selecting all available sets guarantees coverage of all elements, which takes $O(s)$, where s is the number of sets. For the graph coloring problem, assigning a unique color to each graph instance provides a trivial feasible solution in $O(g)$, where g is the number of graphs.

Comparison. We compare Rasengan with previous VQAs that support constrained binary optimization, including penalty-terms-based QAOA (P-QAOA)[39], commute-Hamiltonian-based QAOA (Choco-Q)[22], and hardware-efficient ansatz (HEA)[24]. For P-QAOA, we integrate it with two state-of-the-art QAOA optimization techniques, *FrozenQubits* [3] and *Red-QAOA* [40]. Specifically, *FrozenQubits* aims to reduce the circuit depth and improve the quality of the solution, while *Red-QAOA* optimizes initial parameters. In Choco-Q, we use the state-of-the-art unitary decomposition technique to decompose the mixer unitary. For HEA, we use the circuit structure provided by Kandala et.al [24]. When designing HEA, we introduce a penalty method to make the output satisfy the constraints as much as possible. For parameter updating, we use the constrained optimization by linear approximation method [33] for all baselines and Rasengan.

Platforms. We conduct several small-scale evaluations on two IBMQ devices, including *Kyiv* platform and *Brisbane* platform with 127-qubit Eagle r3 type [11]. The simulation experiments and the classical part are executed on an AMD EPYC 9554 64-core server with

1T RAM. Since QAOA and HEA contain Rotation-X gates, we accelerate their simulation on one A100 GPU by CDUA-quantum [25]. Circuits of Rasengan only include X, control-X, and phase gates, so we accelerate their simulation on the DDSim simulator [7, 45].

Evaluation metrics. Following the approach of prior constrained optimization solvers [39, 40], we employ the *approximation ratio gap (ARG)* as an evaluation metric to measure the gap between solutions generated by the algorithm and the optimal one. ARG is defined as follows:

$$ARG = \left| \frac{E_{opt} - E_{real}}{E_{opt}} \right| \quad (9)$$

where E_{opt} is the objective function value of the optimal solution, and E_{real} is the objective function value of the solution generated by the algorithm. Note that $ARG \in [0, +\infty)$, with lower values indicating better solution quality. We also consider the in-constraints rate to evaluate the ability to incorporate the constraints. The in-constraints rate is the probability that the output solutions satisfy the constraints. When implementing on real-world devices, we further evaluate the latency, including the compilation time for Hamiltonian decomposition, circuit execution time, and the parameter updating time for iterative optimization.

5.2 Algorithmic Evaluation

In Table 2, we compare ARG, circuit depth, and the number of parameters in a noise-free environment. We set the number of repeated layers to five for all baselines, including HEA, P-QAOA, and Choco-Q, to enhance solving accuracy. The maximum number of

iterations of parameter optimization is set to 300 to allow the optimization process to converge to the lowest cost value. In conclusion, Rasengan demonstrates significant advantages in ARG and circuit depth across benchmarks. Though Rasengan requires 0.06 \times more parameters than P-QAOA and Choco-Q, it is still significantly fewer than HEA, ensuring its scalability and applicability in real-world quantum hardware.

Evaluation on ARG. First of all, Rasengan achieves the smallest ARG across benchmarks, highlighting the ability to adapt to diverse problem domains. Specifically, it reduces ARG by 1954 \times and 1897 \times compared to conventional approaches like HEA and P-QAOA respectively, even by 4.12 \times compared to the best baseline (Choco-Q). For small-size problems, Rasengan achieves ARG values as low as 0.01 (F1), 0.07 (K1), and 0.01 (J1), outperforming HEA and P-QAOA by orders of magnitude. Compared to Choco-Q, Rasengan performs particularly well on GCP, which reduces ARG to 0.15 (G1) and 0.93 (G4), while commute-based QAOA yields higher values.

Evaluation on circuit depth. While providing high-quality solutions, Rasengan also achieves the smallest execution circuit depth. Overall, Rasengan achieves around 1.96 \times ~ 49.0 \times reduction of circuit depth compared to baselines. In particular, for FLP benchmarks, Rasengan achieves depths of 34 (F1) and 58 (F4), while Choco-Q has a depth of 3848 (F4). Such a reduction by up to 49 \times demonstrates Rasengan's deployability, particularly in today's NISQ devices that require shallow circuits to mitigate noise and errors. We can observe that the circuit depth for larger-scale problems does not increase and even decreases in some benchmarks. This is because the number of homogeneous basis vectors will grow with the problem scale. Rasengan employs Hamiltonian simplification by basis transformation. More homogeneous basis vectors mean more optimization opportunities, significantly reducing the circuit depth of Rasengan.

Evaluation on the number of parameters. The number of parameters represents the cost of classical computation, that is, the complexity of optimizing the parameters, which is usually polynomial [44]. As listed in Table 2, the number of parameters of HEA is over 10 \times than the other three algorithms. This is because the HEA applies as many parameters as possible to a single bit to achieve the ability to characterize information within a shallow circuit. However, the other three assign parameters based on the number of Hamiltonian simulations. Since we set P-QAOA and Choco-Q to use five layers and each layer constrains two Hamiltonians, the number of their parameters is always 10. Although Rasengan uses more parameters in certain cases like the set covering problem, our design achieves a good balance between the number of parameters and solution accuracy, which is crucial for the practical scalability of quantum computing.

Simulations with deeper QAOA layers. Rasengan features a fixed circuit depth. While theoretically, QAOA can achieve better algorithmic performance by increasing the number of layers [15]. Figure 9 compares QAOAs using different numbers of layers on the F1 benchmark. The ARG of Choco-Q is close to Rasengan after 14 layers. However, the 14-layer Choco-Q necessitates a circuit depth of 1419. On the other hand, Rasengan only takes 3 segments, with each segment requiring 49 circuit depth. P-QAOA shows limited improvement even with more layers. This is because the penalty term can increase the non-convexity of the loss function, making

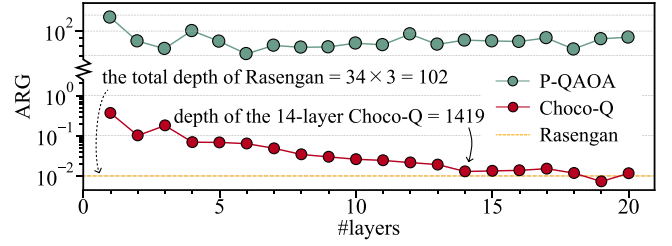


Figure 9: Evaluation of ARG using the different number of QAOA layers.

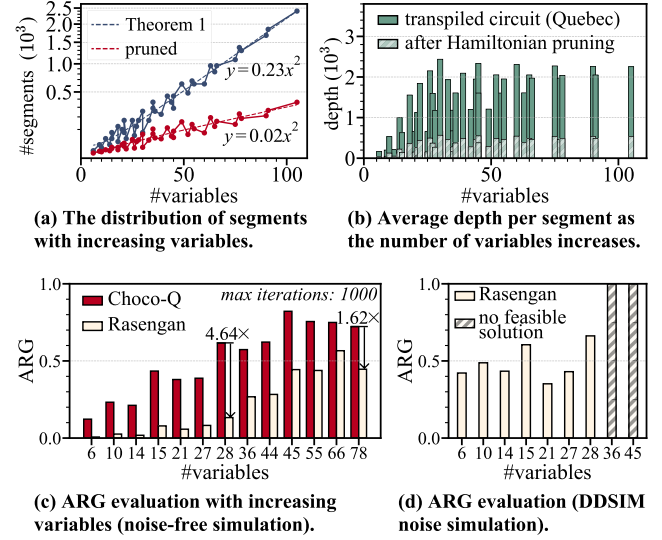


Figure 10: Scalability analysis on large-scale FLP problems.

the optimization more prone to local optima. These results demonstrate that Rasengan outperforms other QAOAs in ARG, even if we introduce deeper QAOA layers.

Application dependency. In Table 2, the number of feasible solutions reflects the complexity of the solution space. Among all benchmarks, SCP exhibits the fastest growth, with S4 having the largest number of feasible solutions (856). On S4, all methods yield higher ARGs exceeding 1, but Rasengan outperforms other VQAs, demonstrating its superior performance in larger solution spaces. Despite KPP having a smaller feasible space, its parameter count is not the lowest. This is because its constraints span more qubits, and the transition Hamiltonian involves the largest number of qubits, making effective transitions harder to match. As a result, within one τ cycle (with m transitions), only a small subset can respond effectively. The strong dependency between Hamiltonian layers also reduces the effectiveness of Hamiltonian pruning.

5.3 Scalability Analysis

Since the solution space generally grows with the problem size, similar trends are observed in the number of segments, circuit depth, and ARG. The FLP problem allows the construction of many gradually increasing cases. Therefore, we evaluate the scalability of Rasengan on large-scale FLP instances, with the number of variables ranging from 6 to 105. To allow more thorough optimization for

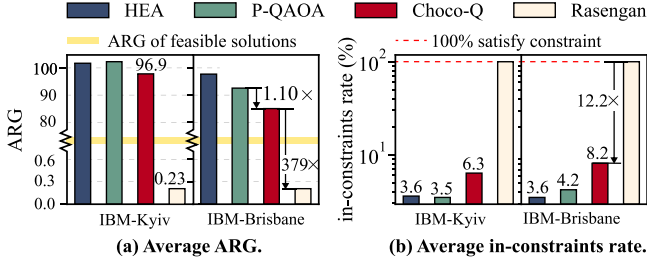


Figure 11: Evaluation on real-world quantum platforms.

larger-scale problems, we set the maximum number of iterations to 1000.

Figure 10 (a) shows that the maximum number of segments follows a quadratic relationship with the number of variables, as proved in Theorem 1. With the application of Hamiltonian pruning, the number of segments can be further reduced. The average circuit depth compiled via Quebec is shown in Figure 10 (b). We can see that the upper bound of circuit depth is around 3×10^3 . This is because segmented execution can maintain a manageable circuit depth, making it a promising approach for large-scale problems. The minimum depth of each segment equals the depth of a single transition Hamiltonian τ simulation circuit. For problems like FLP and SCP, the number of qubits controlled by τ is constant due to fixed-size constraints. As a result, the segment depth tends to remain steady as the problem scales. In contrast, for problems like GCP, τ involves more qubits as the problem grows, leading to an increasing segment depth.

To further evaluate Rasengan on larger-scale FLP problems, we conduct both noise-free and noisy simulations. The noise-free results are shown in Figure 10 (c). When solving an 78-qubit problem, Rasengan achieves lower ARGs than 0.5, while Choco-Q on a 28-qubit problem has a higher ARG than 0.5. The larger scalability of Rasengan demonstrates the advantage of the expansion-based design. Figure 10 (d) shows the results under noise. When the problem size exceeds 28 qubits, some segments fail to produce feasible solutions due to high error rates. As a result, no valid state is available for initializing the next segment, and the optimization terminates early. However, when the segment depth remains shallow enough to yield feasible solutions, Rasengan continues to perform effectively.

5.4 Evaluation on Real-world Platforms

We evaluate Rasengan on the IBM cloud platform with Kyiv and Brisbane devices. Considering the large circuit depth of baselines and the short decoherence time of these devices, we chose to implement the small-scale problems, including F1, K1, and J1 in Table 2. We employ the same settings as Section 5.2. Considering the operational cost of using IBM's real quantum computers, we set a maximum of 100 iterations. In practice, this setting is sufficient to ensure the convergence of the algorithms.

Average ARG on different devices. Figure 11 (a) shows the average ARG of different algorithms on IBM-Kyiv and IBM-Brisbane devices. Overall, Rasengan outperforms all prior works, achieving at least 379 \times improvement. The ARG of prior works is even higher than the average ARG of all feasible solutions, which means they output too many unsatisfiable results in a noisy environment. The

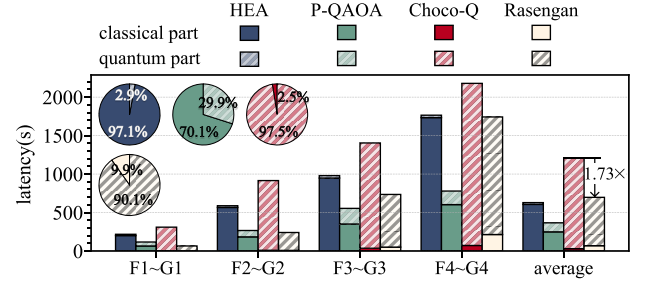


Figure 12: Latency breakdown of different methods.

circuit depth of Rasengan is less than 100, making Rasengan more deployable in NISQ devices. Besides, Rasengan employs purification-based error mitigation, which can ensure the output is always inside the constraints, avoiding invalid searching outside of the constraints. The two-qubit gate error rate of IBM-Kyiv is 1.2%, which is 1.48 \times lower than IBM-Brisbane (0.82%). This difference aligns with the results in Figure 11 (a), where the ARG of baselines on IBM-Brisbane is relatively lower than that on IBM-Kyiv. However, Rasengan is not afraid of the effects of increased noise. Even though the error rate has increased by 1.48 \times , the ARG of Rasengan has no visible change. This result reveals our robust performance on different scales of noise.

Average in-constraints rate on different devices. The average in-constraints rate of different algorithms is represented in Figure 11 (b). Obviously, by leveraging the error mitigation with purification, Rasengan achieved a 100% in-constraints rate in noisy devices. Although Choco-Q states that it can achieve a 100% in-constraints rate theoretically, the practical in-constraints rate has declined to 6.3% in the IBM-Kyiv device. Even though the in-constraints rate of baselines is increased in a less noisy device like IBM-Brisbane, Rasengan still holds an advantage with a 12.2 \times improvement.

Training latency breakdown. Figure 12 presents the training latency, including the classical and quantum parts. The classical time of HEA and P-QAOA takes over 70% of the total latency because they cannot guarantee solutions under the constraints, requiring more calculations of objective functions for the solutions outside the constraints. Besides, the quadratic terms in the penalty function also increase the classical latency. Compared to Choco-Q, Rasengan reduces the total time by 1.73 \times . The classical latency of Rasengan is slightly higher than that of Choco-Q. This is because Rasengan leverages segmented execution, which requires classical calculation. However, in the NISQ era, quantum operations such as circuit execution and measurement remain a major bottleneck. Reducing quantum overhead is therefore critical for improving training efficiency.

Overhead of segmented execution. To evaluate the overhead of segmented execution, Figure 13 depicts the shots and latency when setting different numbers of segments. In our experiments, the number of shots is set to 1024 per segment. We observe a linear relationship between shots and the number of segments in Figure 13 (a) since 1024 shots provide sufficient accuracy at this scale, the additional shots required for each incremental segment remain constant. Figure 13 (b) demonstrates a sub-linear relationship

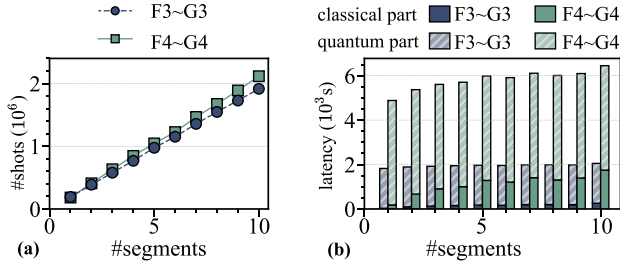


Figure 13: Shots and latency of Rasengan with different numbers of segments.

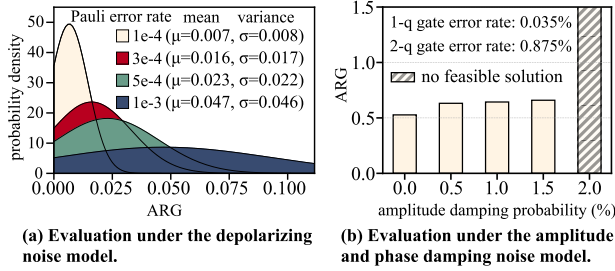


Figure 14: Evaluation on different noise models.

between latency and the number of segments, mainly due to classical computation overhead. To be specific, segmented execution requires handling intermediate measurements, which introduces additional classical calculations. However, this classical overhead is acceptable considering the finite number of measured basis states. On the other hand, the quantum latency increases slightly with more segments thanks to the constant circuit depth. The additional latency from measurements and initialization is negligible. Given that quantum execution time dominates the overall latency, segmented execution does not fundamentally limit the scalability of Rasengan.

5.5 Noise Sensitivity Analysis of Rasengan

Sensitivity to depolarizing noise. We evaluate Rasengan’s noise sensitivity under different error rates based on the Pauli noise model. Figure 14 (a) depicts the ARG distribution of Rasengan under different Pauli error rates, where the data are collected from 2000 problems. The error rate is derived from the data of IBM devices [11] and is mostly on the order of 10^{-3} . When the error rate is set to 1‰ , more than 99% of ARGs are less than 0.025. Though the average ARG increases with the error rates, it is still less than 0.15 with a 0.001 error rate, indicating that Rasengan can still output a better solution with a relatively high probability.

Sensitivity to amplitude damping. Figure 14 (b) shows the ARG under increasing amplitude damping noise, with fixed background noise consisting of phase damping and depolarizing noise configured according to calibration data from IBMQ devices[11]. The single-qubit and two-qubit gate error rates are set to 0.035% and 0.875%, respectively. As the amplitude damping probability increases from 0.0% to 1.5%, the ARG slightly increases, indicating the resilience of the Rasengan optimizer. However, when the damping probability reaches 2%, intermediate states from segmented

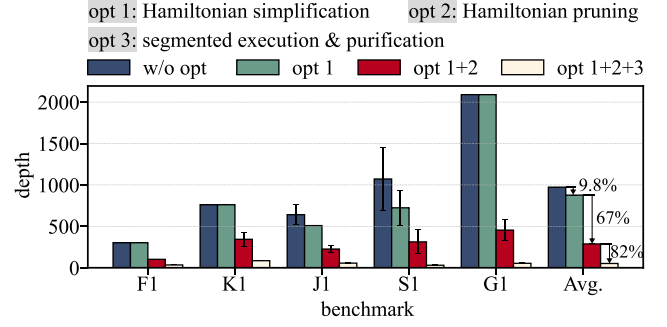


Figure 15: Ablation study of optimization strategies on circuit depth.

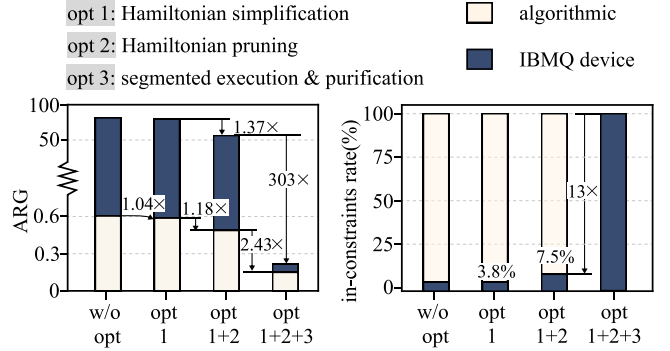


Figure 16: Ablation study on ARG and in-constraints rate.

execution frequently become infeasible. Since infeasible intermediate states cannot be passed to the next segment, the optimization process terminates, and no valid solution can be found.

5.6 Ablation Study of Optimization Techniques

We study the effect of different optimization techniques in Rasengan, including transition Hamiltonian simplification (opt 1) and pruning (opt 2), probability-preserving segmented execution & error mitigation by purification (opt 3). The evaluated metric involves circuit depth, ARG, and in-constraints rate.

Ablation study on circuit depth. Figure 15 depicts the circuit depth after applying different optimization strategies. On average, the three optimization strategies achieved circuit depth reductions of 9.8%, 67%, and 82%, respectively. Opt 1 simplifies transition Hamiltonians by generating a lightened basis but is ineffective for F1, K1, and G1, where constraints are already in their sparsest form. Opt 2 reduces circuit depth by over 50%, revealing widespread redundancy in real-world problems. Pruning redundant states achieves significant depth reduction. Opt 3 further reduces circuit depth by 82% by segmenting transition Hamiltonian sequences on top of opt 1 and opt 2, delivering the most substantial improvement.

Ablation study on ARG and in-constraints rate. Using the same cases as in Section 5.4, we evaluate the impact of optimization strategies on both ARG and in-constraints rate across noise-free simulators and IBM quantum devices (IBM-Kyiv and IBM-Brisbane), as shown in Figure 16. For ARG (left), Opt 1 yields limited improvement (1.04 \times) since regenerating the basis does not alter the overall evolution process. Opt 2 improves ARG by 1.18 \times on simulators and

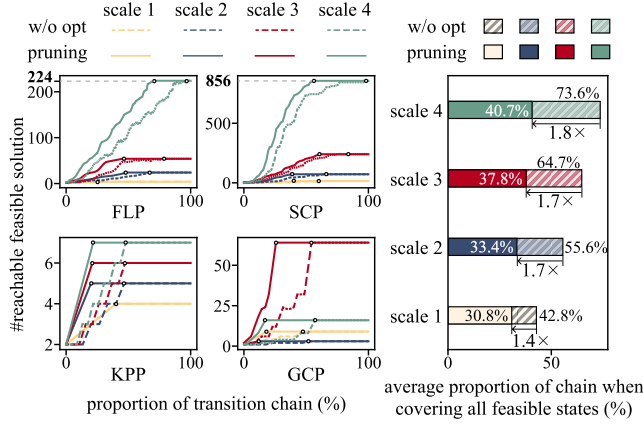


Figure 17: Solution space analysis on Hamiltonian Pruning.

1.37× on hardware, benefiting from a pruning strategy that reduces invalid transitions and shortens circuit depth by 67%. Opt 3 further boosts ARG by 2.43×, with segmented execution preserving the probability distribution and avoiding reversals in superposition. The purification strategy then brings a dramatic 303× improvement on hardware. For in-constraints rate (right), Rasengan achieves 100% at the algorithmic level by operating strictly within the feasible solution space. In noisy settings, low in-constraints rates are mainly due to deep circuits. Opt 2 alleviates this by reducing depth, improving the rate from 3.8% to 7.5%, while Opt 3, through purification-based error mitigation, achieves the most significant enhancement and ensures constraint satisfaction in the final output.

5.7 Evaluation on Hamiltonian Pruning

We study the effect of the transition Hamiltonian pruning technique in accelerating search space expansion within the FLP, KPP, SCP, and GCP. Figure 17 illustrates the acceleration in search space expansion speed achieved by pruning, compared to the original unpruned transition chains. The proportion of pruned chains is measured relative to the total length of the chains. Across the four scales, pruning consistently accelerates search space expansion. For the fourth scale, the unpruned chains require 73.6% of the total chain length to cover the entire feasible solution space, whereas the pruned chains achieve the same coverage with only 40.7% of the total chain length, improving the expansion speed by 1.8×. In GCP, the total number of feasible solutions on the third scale is larger than that on the fourth, and the first scale has more feasible solutions than the second. This is because, in GCP, both the number of variables and the number of constraints increase with scale. The additional constraints reduce the size of the feasible solution space. This trend is specific to the selected graph instances and varies with different topologies.

6 Related work

Many quantum algorithms have been proposed to solve binary optimization problems with computational advantages. For unconstrained problems, quantum annealing [34, 37] is the first, that utilizes the quantum adiabatic theorem. However, quantum annealing faces significant limitations, including long embedding and

evolution times [28]. To overcome these issues, QAOA [15, 41, 44] offers a gate model for quantum annealing, employing a variational quantum algorithm to shorten evolution time. Since QAOA can be deployed on noisy quantum devices and has been optimized in aspects such as compilation [1], parameter initialization [40], parameter updating [36], and distributed versions [3], it has gained widespread attention. However, both quantum annealing and QAOA struggle to incorporate constraints effectively.

Several approaches have been proposed to address the challenges of handling constraints. Commute-Hamiltonian-based QAOA replaces the original mixer in QAOA with a Hamiltonian that commutes with constraints to embed constraints. However, it suffers deep circuits and high sensitivity to quantum noise. Q-CHOP [19] constructs a continuous Hamiltonian path that keeps the system within a feasible solution space but cannot guarantee a 100% in-constraint rate. Similarly, Franz G. Fuchs et al. [17] extended the Hamiltonian to support various types of constraints, but this method requires prior knowledge of all possible solutions, leading to high computational costs.

Additionally, several universal quantum algorithms have been developed by building upon quantum annealing and QAOA with specialized modifications. One such method is the hardware-efficient ansatz (HEA) [10, 24], where each variable is encoded by a single qubit, and the objective function is modified similarly to penalty-QAOA. Although hardware-efficient, this method may not always converge to the optimal solution due to the unspecialized circuit structure. Another approach, Grover adaptive search, utilizes the Grover algorithm [30] with a selection circuit to exclude solutions outside the constraints [18]. However, the complexity of the selection circuit limits its implementation on current hardware, and the search process generates many invalid solutions, requiring extensive iterations to find the desired solution.

7 Conclusion

We propose Rasengan, a high-precision and hardware-efficient approach for constrained binary optimization problems, via rigorously designed transition Hamiltonian that can cover the quantum superposition of all feasible solution states. Then, we develop three optimization techniques for efficient and accurate Rasengan implementation. Specifically, we compressed the circuit complexity of transition Hamiltonian by simplification and pruning. We also segment the quantum circuit into pieces with limited depth while preserving the probability, catering to the limited reliable execution time of quantum devices. Based on the measurement results from circuit segments, we timely mitigate the device by constraint-based purification, avoiding execution errors accumulated in the final results. Overall, these three optimizations work seamlessly with the transition Hamiltonian in Rasengan, shrinking the circuit depth to a deployable level and thereby enabling a substantial enhancement in the ARG for solving constrained optimization problems.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No.2023YFF0905200) and National Natural Science Foundation of China (No.62472374). This work was also funded by Zhejiang Pioneer (Jianbing) Project (No.2023C01036).

8 Artifact Appendix

8.1 Abstract

This artifact contains all necessary code and scripts to reproduce the key experimental results of the paper, including Figures 9–17 and Table 2. It provides Jupyter notebooks and Python programs for running and visualizing experiments, along with a consolidated batch script and a summary notebook for streamlined reproduction and inspection.

8.2 Artifact check-list (meta-information)

- **Algorithm:** Rasengan
- **Run-time environment:** Linux, Python, Jupyter
- **Hardware:** Dual AMD EPYC 9554 CPUs, NVIDIA H100 GPU (optional)
- **Execution:** Qiskit circuit simulation
- **How much disk space is required (approximately)?:** Less than 5 GB to store the artifact directory and Python virtual environment
- **How much time is needed to prepare workflow (approximately)?:** 10 minutes
- **How much time is needed to complete experiments (approximately)?:** 40 hours on dual AMD EPYC 9554 CPUs (128 cores total); 25 hours with GPU acceleration
- **Publicly available?:** Yes
- **Code licenses (if publicly available)?:** MIT License
- **Archived (provide DOI)?:** 10.5281/zenodo.16730424

8.3 Description

Rasengan is implemented as a collection of Python scripts and Jupyter notebooks, which can be executed on Linux systems. Below, we introduce the important files and directories in the artifact, highlighting their purpose for reproducing the experimental results.

Repository structure:

- `env_setup/`: Contains Conda environment YAML files for both CPU-only and optional GPU setups, enabling reproducible software environments.
- `rasengan/`: Implements the Rasengan algorithm, including all core modules for generating and simulating transition-Hamiltonian-based quantum circuits.
- `reproduce/`: Provides scripts and notebooks to reproduce the experiments from the paper. Each experiment (corresponding to Figures 9–17 and Table 2) resides in a separate subdirectory.

How to access:

GitHub: <https://github.com/JanusQ/rasengan>
DOI: <https://doi.org/10.5281/zenodo.16730424>

Hardware dependencies:

The artifact relies on Qiskit-based quantum circuit simulation. It is CPU-compatible and optionally supports GPU acceleration via the `qiskit-aer-gpu` backend.

Software dependencies:

The code requires Python 3.10 or higher. All dependencies are listed in the Conda YAML file in `env_setup/`.

Reproduction scaling:

The original evaluation used 20 benchmark problems with 100 cases each, which may require several days of computation. For reproducibility within a reasonable time, the provided scripts scale down the number of benchmarks and reduce the number of cases per benchmark from 100 to approximately 10. This modification significantly lowers the computational cost, but may introduce slight statistical bias compared to the original results.

8.4 Installation

Installation instructions are provided in `README.md`. Below is a summary.

Create environment:

```
cd env_setup
conda env create -f cpu-env.yml
conda activate rasengan
```

(Optional) Enable GPU acceleration:

```
pip install qiskit-aer-gpu
```

Verify installation:

```
python env_check.py
```

Additional notes on GPU support:

Rasengan already runs efficiently on CPU. GPU acceleration via `qiskit-aer-gpu` is optional and mainly benefits baseline simulations such as HEA and QAOA, which involve many RX gates. The programs automatically detect GPU availability, and no manual switching is required.

8.5 Evaluation and Expected Results

Each experiment subdirectory under `reproduce/` (e.g., `figure_x/`, `table_2/`) contains:

- `run_and_plot.ipynb`: The main notebook to run and visualize the experiment.
- `run_backend.py`: Batch execution script for headless runs.
- `only_plot.ipynb`: A notebook to generate plots from cached results.

Two recommended workflows:

- (1) Run all experiments in one go:
`python reproduce/run_all_experiments.py`
- (2) Run experiments individually from each subdirectory using either the notebook or the backend script.

Results are automatically saved in the correct subdirectories. By default, rerunning a script will overwrite existing output files unless cached results are reused. Aggregated comparisons are available in `reproduce/results_summary.ipynb`.

8.6 Methodology

Submission, reviewing, and badging methodology:

- <https://www.acm.org/publications/policies/artifact-review-badging>
- <http://cTuning.org/ae/submission-20201122.html>
- <http://cTuning.org/ae/reviewing-20201122.html>

References

- [1] Mahabubul Alam, Abdullah Ash-Saki, and Swaroop Ghosh. 2020. An efficient circuit compilation flow for quantum approximate optimization algorithm. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 1–6.
- [2] Khulood Alyahya and Jonathan E. Rowe. 2019. Landscape Analysis of a Class of NP-Hard Binary Packing Problems. *Evolutionary Computation* 27, 1 (03 2019), 47–73.
- [3] Ramin Ayanzadeh, Narges Alavisamani, Poulami Das, and Moinuddin Qureshi. 2023. 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)*. 311–324.
- [4] Mayowa Ayodele. 2022. Penalty weights in qubo formulations: Permutation problems. In *European Conference on Evolutionary Computation in Combinatorial Optimization (EvoStar)*. 159–174.
- [5] Sebastian Brandhofer, Daniel Braun, Vanessa Dehn, Gerhard Hellstern, Matthias Hüls, Yanjun Ji, Ilia Polian, Amandeep Singh Bhatia, and Thomas Wellens. 2023. Benchmarking the performance of portfolio optimization with QAOA. *Quantum Inf. Process.* 22, 25 (2023).
- [6] Thang Nguyen Bui and Byung Ro Moon. 1996. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers (TC)* 45, 7 (1996), 841–855.
- [7] Lukas Burgholzer, Alexander Ploier, and Robert Wille. 2023. Simulation Paths for Quantum Circuit Simulation With Decision Diagrams What to Learn From Tensor Networks, and What Not. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 42, 4 (2023), 1113–1122.
- [8] Alberto Caprara, Paolo Toth, and Matteo Fischetti. 2000. Algorithms for the set covering problem. *Annals of Operations Research* 98, 1 (2000), 353–371.
- [9] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J Coles. 2021. Variational quantum algorithms. *Nature Reviews Physics* 3, 9 (2021), 625–644.
- [10] Siddharth Dangwal, Gokul Subramanian Ravi, Poulami Das, Kaitlin N Smith, Jonathan Mark Baker, and Frederic T Chong. 2023. VarSaw: Application-tailored Measurement Error Mitigation for Variational Quantum Algorithms. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 362–377.
- [11] IBM Quantum Devices. 2024. ibmq quito. <https://quantum-computing.ibm.com/services/resources>
- [12] Paul A. M. Dirac. 1930. *The Principles of Quantum Mechanics* (1st ed.). Oxford University Press, Oxford, UK.
- [13] Heiko Falk. 2009. WCET-aware register allocation based on graph coloring. In *Proceedings of the 46th Annual Design Automation Conference (DAC)*. 726–731.
- [14] Reza Zanjirani Farahani and Masoud Hekmatfar. 2009. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media.
- [15] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028* (2014).
- [16] David Fitzek, Toheed Ghandriz, Leo Laine, Mats Granath, and Anton Frisk Kockum. 2024. Applying quantum approximate optimization to the heterogeneous vehicle routing problem. *Sci. Rep.* 14, 25415 (2024).
- [17] Franz Georg Fuchs, Kjetil Olsen Lye, Halvor Møll Nilsen, Alexander Johannes Stasik, and Giorgio Sartor. 2022. Constraint preserving mixers for the quantum approximate optimization algorithm. *Algorithms* 15, 6 (2022), 202.
- [18] Austin Gilliam, Stefan Woerner, and Constantin Goniculea. 2021. Grover adaptive search for constrained polynomial binary optimization. *Quantum* 5 (2021), 428.
- [19] Pranav Gokhale. 2022. Q-CHOP: Quantum Constrained Hamiltonian Optimization. In *APS March Meeting Abstracts*, Vol. 2022. K36–008.
- [20] TM Graham, Y Song, J Scott, C Poole, L Phuttitarn, K Jooya, P Eichler, X Jiang, A Marra, B Grinkemeyer, M Kwon, M Ebert, J Cherek, MT Lichtman, M Gillette, J Gilbert, D Bowman, T Ballance, C Campbell, ED Dahl, O Crawford, NS Blunt, B Rogers, T Noel, and M Saffman. 2022. Multi-qubit entanglement and algorithms on a neutral-atom quantum computer. *Nature* 604, 7906 (2022), 457–462.
- [21] Werner H Greub. 2012. *Linear algebra*. Vol. 23. Springer Science & Business Media.
- [22] Itay Hen and Marcelo S Sarandy. 2016. Driver Hamiltonians for constrained optimization in quantum annealing. *Physical Review A* 93, 6 (2016), 062312.
- [23] Tommy R Jensen and Bjarne Toft. 2011. *Graph coloring problems*. John Wiley & Sons.
- [24] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. 2017. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature* 549, 7671 (2017), 242–246.
- [25] Jin-Sung Kim, Alex McCaskey, Bettina Heim, Manish Modani, Sam Stanwyck, and Timothy Costa. 2023. CUDA Quantum: The Platform for Integrated Quantum-Classical Computing. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. 1–4.
- [26] A Yu Kitaev. 1995. Quantum measurements and the Abelian stabilizer problem. *arXiv preprint quant-ph/9511026* (1995).
- [27] Dušan Knop, Michał Pilipczuk, and Marcin Wrochna. 2020. Tight complexity lower bounds for integer linear programming with few constraints. *ACM Transactions on Computation Theory (TCT)* 12, 3 (2020), 1–19.
- [28] Christopher R Laumann, Roderich Moessner, Antonello Scardicchio, and Shivaaji Lal Sondhi. 2015. Quantum annealing: The fastest route to quantum computation. *The European Physical Journal Special Topics* 224, 1 (2015), 75–88.
- [29] Hannes Leipold and Federico M Spedalieri. 2021. Constructing driver Hamiltonians for optimization problems with linear constraints. *Quantum Science and Technology* 7, 1 (2021), 015013.
- [30] Gui-Lu Long. 2001. Grover algorithm with zero theoretical failure rate. *Physical Review A* 64, 2 (2001), 022307.
- [31] Antón Makarov, Carlos Pérez-Herradón, Giacomo Franceschetto, Márcio M. Taddei, Eneko Osaba, Paloma del Barrio Cabello, Esther Villar-Rodríguez, and Izaskun Oregi. 2024. Quantum Optimization Methods for Satellite Mission Planning. *IEEE Access* 12 (2024), 71808–71820.
- [32] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O’Brien. 2014. A variational eigenvalue solver on a photonic quantum processor. *Nature communications* 5, 1 (2014), 4213.
- [33] Michael JD Powell. 1994. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer.
- [34] Giuseppe E Santoro and Erio Tosatti. 2006. Optimization using quantum mechanics: quantum annealing through adiabatic evolution. *Journal of Physics A: Mathematical and General* 39, 36 (2006), R393.
- [35] Peter W Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 124–134.
- [36] Michael Streif and Martin Leib. 2020. Training the quantum approximate optimization algorithm without access to a quantum processing unit. *Quantum Science and Technology* 5, 3 (2020), 034008.
- [37] Siwei Tan, Mingqian Yu, Andre Python, Yongheng Shang, Tingting Li, Liqiang Lu, and Jianwei Yin. 2023. HyQSAT: A Hybrid Approach for 3-SAT Problems by Integrating Quantum Annealer with CDCL. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 731–744.
- [38] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. 2022. The Variational Quantum Eigensolver: A review of methods and best practices. *Physics Reports* 986 (2022), 1–128. The Variational Quantum Eigensolver: a review of methods and best practices.
- [39] Amit Verma and Mark Lewis. 2022. Penalty and partitioning techniques to improve performance of QUBO solvers. *Discrete Optimization* 44 (2022), 100594.
- [40] Meng Wang, Bo Fang, Ang Li, and Prashant J Nair. 2024. 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)*. 980–998.
- [41] Sha-Sha Wang, Hai-Ling Liu, Yong-Mei Li, Fei Gao, Su-Juan Qin, and Qiao-Yan Wen. 2024. Variational Quantum Algorithm-Preserving Feasible Space for Solving the Uncapacitated Facility Location Problem. *Advanced Quantum Technologies* (2024), 2400201.
- [42] Wikipedia contributors. 2024. Identical-machines scheduling — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Identical-machines_scheduling [Online; accessed 12-October-2024].
- [43] Debin Xiang, Qifan Jiang, Liqiang Lu, Siwei Tan, and Jianwei Yin. 2025. Choco-Q: Commute Hamiltonian-based QAOA for Constrained Binary Optimization. In *2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*.
- [44] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. 2020. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X* 10, 2 (2020), 021067.
- [45] Alwin Zulehner and Robert Wille. 2019. Advanced Simulation of Quantum Computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 5 (2019), 848–859.