

Hierarchical QAOA circuit design framework for distributed quantum computing

Ting-Yu Luo(骆挺宇), and Yu-Xin Deng(邓玉欣)
Citation: Chin. Phys. B, 2026, 35 (**4**): 040309. DOI: 10.1088/1674-1056/ae00ae
http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Distributed Kuperberg's algorithm

Peng-Yu Yang(杨鹏宇), Xin Zhang(张新), and Song Lin(林崧)
Chin. Phys. B, 2026, 35 (**3**): 030303. DOI: 10.1088/1674-1056/adfb58

Enhancing the performance of quantum battery by squeezing reservoir engineering

Yue Li(李月), Rong-Fang Liu(刘蓉芳), Jia-Bin You(游佳斌), Wan-Li Yang(杨万里), and Hua Guan(管桦)
Chin. Phys. B, 2026, 35 (**1**): 010303. DOI: 10.1088/1674-1056/ae1df2

Distributed quantum circuit partitioning and optimization based on combined spectral clustering and search tree strategies

Zilu Chen(陈子禄), Zhijin Guan(管致锦), Shuxian Zhao(赵书娴), and Xueyun Cheng(程学云)
Chin. Phys. B, 2025, 34 (**5**): 050305. DOI: 10.1088/1674-1056/adbada

Automatic architecture design for distributed quantum computing

Ting-Yu Luo(骆挺宇), Yu-Zhen Zheng(郑宇真), Xiang Fu(付祥), and Yu-Xin Deng(邓玉欣)
Chin. Phys. B, 2024, 33 (**12**): 120302. DOI: 10.1088/1674-1056/ad7c2c

Quafu-Qcover: Explore combinatorial optimization problems on cloud-based quantum computers

Hong-Ze Xu(许宏泽), Wei-Feng Zhuang(庄伟峰), Zheng-An Wang(王正安), Kai-Xuan Huang(黄凯旋), Yun-Hao Shi(时运豪), Wei-Guo Ma(马卫国), Tian-Ming Li(李天铭), Chi-Tong Chen(陈驰通), Kai Xu(许凯), Yu-Long Feng(冯玉龙), Pei Liu(刘培), Mo Chen(陈墨), Shang-Shu Li(李尚书), Zhi-Peng Yang(杨智鹏), Chen Qian(钱辰), Yu-Xin Jin(靳羽欣), Yun-Heng Ma(马运恒), Xiao Xiao(肖骁), Peng Qian(钱鹏), Yanwu Gu(顾炎武), Xu-Dan Chai(柴绪丹), Ya-Nan Pu(普亚南), Yi-Peng Zhang(张翼鹏), Shi-Jie Wei(魏世杰), Jin-Feng Zeng(增进峰), Hang Li(李行), Gui-Lu Long(龙桂鲁), Yirong Jin(金贻荣), Haifeng Yu(于海峰), Heng Fan(范桁), Dong E. Liu(刘东), and Meng-Jun Hu(胡孟军)
Chin. Phys. B, 2024, 33 (**5**): 050302. DOI: 10.1088/1674-1056/ad18ab

---

Hierarchical QAOA circuit design framework for distributed quantum computing

Ting-Yu Luo(骆挺宇)[1] and Yu-Xin Deng(邓玉欣)[1,2,†]

[1] *Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai* 200062, *China*
[2] *School of Computing and Artificial Intelligence, Shanghai University of Finance and Economics, Shanghai* 200433, *China*

(Received 25 June 2025; revised manuscript received 13 August 2025; accepted manuscript online 29 August 2025)

The quantum approximate optimization algorithm (QAOA) is a promising approach for solving combinatorial optimization problems on real quantum devices. As QAOA scales to tackle larger problem instances, the limited qubit capacity of single-chip systems becomes a critical bottleneck. To overcome this limitation, distributed quantum computing (DQC) provides a scalable solution. However, when QAOA circuits are executed in such systems, their performance is significantly hindered by the high cost of remote communication. Motivated by this challenge, we propose *HiQ-DF*, a QAOA circuit design framework tailored for DQC systems. By employing a hierarchical optimization strategy, *HiQ-DF* enables comprehensive multi-objective optimization during circuit construction. Experimental results on QAOA circuits solving MaxCut instances show that our framework significantly outperforms baseline methods, achieving an average reduction of 26.12% in EPR pair usage (up to 36.85%), 26.44% in circuit latency (up to 35.27%), and 39.63% in circuit depth (up to 49.3%).

Keywords: quantum circuit design, quantum approximate optimization algorithm, distributed quantum computing

PACS: 03.67.Lx, 03.67.Ac, 85.25.–j

DOI: 10.1088/1674-1056/ae00ae **CSTR:** 32038.14.CPB.ae00ae

1. Introduction

Quantum computing has emerged as a promising paradigm for solving computational challenges that exceed the capabilities of classical computers.[1–4] Among its applications, combinatorial optimization is especially significant, with real-world relevance in logistics,[5] finance,[6] and machine learning.[8] To tackle such problems, the quantum approximate optimization algorithm (QAOA)[8] offers a hybrid quantum–classical framework well-suited to noisy intermediate-scale quantum (NISQ) devices.[9–13] As QAOA scales to more complex instances, the growing demand for high-quality qubits quickly outpaces the capacity of individual chips, which are limited by factors such as fabrication[14] and control challenges.[15] Distributed quantum computing (DQC),[16–18] which interconnects multiple quantum chips into a quantum chip network (QCN), provides a scalable solution to this bottleneck. Nevertheless, deploying QAOA on quantum devices presents several challenges. Even in single-chip systems, the performance of QAOA circuits is already constrained by factors such as circuit depth[11] and noise accumulation,[19] motivating various compilation-level optimization efforts.[20–22] Accordingly, in DQC systems, it is essential to develop dedicated optimization techniques to support efficient QAOA execution.

In DQC, quantum communication overhead critically affects the execution quality of quantum circuits. Specifically, executing two-qubit gates between qubits located on different chips (i.e., remote quantum gates) typically requires the consumption of Einstein–Podolsky–Rosen (EPR) pairs,[23] whose preparation latency is approximately 12 times that of local two-qubit gates[24] For QAOA, which heavily relies on two-qubit gates, the impact of quantum communication costs is exceptionally pronounced. Our experimental results reveal that once QAOA circuits are mapped onto QCNs, the proportion of remote quantum gates becomes significantly high. As illustrated in Fig. 1(a), these remote quantum gates account for up to 83% of all two-qubit gates on average. This heavy reliance on remote gates incurs considerable EPR pair consumption, which not only prolongs circuit execution time but also amplifies noise accumulation. Consequently, to enhance QAOA performance on distributed quantum devices, optimization techniques need to strongly prioritize minimizing EPR pair usage.

The existing research on improving the performance of QAOA covers multiple directions, including ansatz circuit design, parameter optimization, hardware adaptation, compilation optimization, and distributed execution. At the ansatz circuit level, numerous studies have tailored QAOA circuits for specific problem instances, particularly by modifying the mixer operator to increase the approximation ratio while reducing circuit depth.[25–27] To address the barren plateau challenge in the QAOA optimization landscape, several works have proposed advanced parameter training strategies.[11,28,29] With respect to hardware adaptation, some

†Corresponding author. E-mail: yxdeng@msg.sufe.edu.cn

© 2026 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

efforts focus on mitigating noise-induced performance degradation on real quantum devices.[30–32] At the compilation level, a range of techniques have been proposed to optimize the implementation of physical gates,[21,31] reduce SWAP gate insertions,[20,33] shorten circuit depth,[25,27] and minimize hardware-related overheads. As problem sizes increase, distributed realizations of QAOA have been investigated to overcome the qubit capacity constraints of single-chip systems and to facilitate large-scale problem solving.[34–36] However, most of these studies focus on scalability through distributed architectures rather than reducing remote quantum communication costs. A few works have attempted to alleviate communication overhead at the compilation stage by leveraging specific hardware configurations in QCN systems,[37] but they do not address circuit-level design optimization tailored for QAOA.

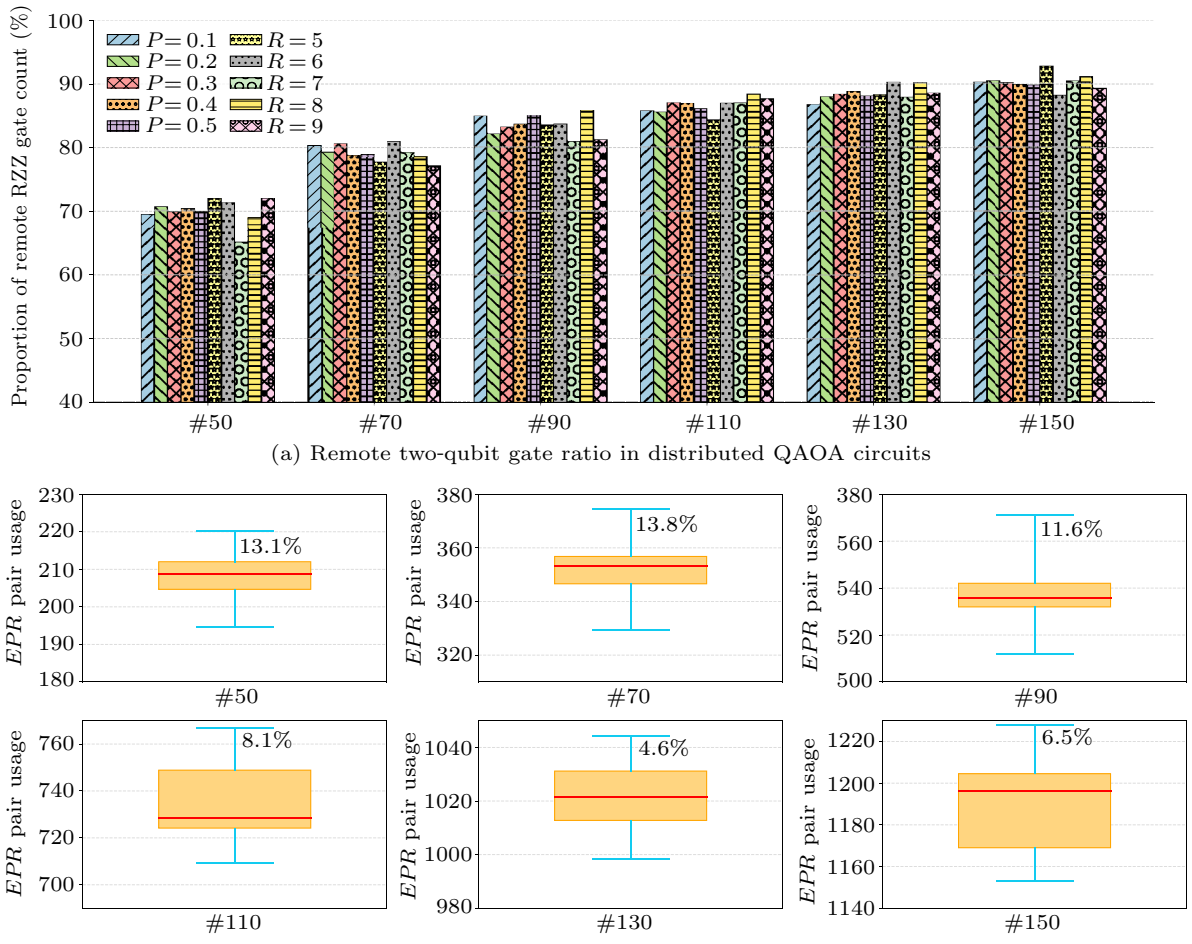


Fig. 1. Proportion of remote two-qubit gates and the effect of randomized gate order on EPR pair usage. For QAOA circuits ($p = 1$, where p is the number of algorithmic layers) solving the MaxCut problem on various regular and random graphs, panel (a) shows the percentage of remote two-qubit gates after mapping the circuits to QCNs. For regular graphs with $R = 5$, where R denotes the degree of each vertex, and varying sizes, each QAOA circuit ($p = 1$) is compiled ten times with randomly reordered two-qubit gates, and the resulting circuits are evaluated. Panel (b) presents the distribution of EPR pair usage, with blue lines indicating the observed minimum and maximum values.

Meanwhile, existing general-purpose compilation techniques for DQC mainly aim to optimize communication overhead for broader quantum algorithms.[28,38–41] However, these approaches are not customized for QAOA. Notably, QAOA exhibits structural properties that could be exploited for more effective optimization. For example, the cost Hamiltonians in each algorithmic layer are often repeated. Nevertheless, such features remain underexploited in existing work. As a result, relying solely on these techniques without incorporating QAOA-specific characteristics makes it difficult to optimize overall quantum communication overhead. Moreover, they fail to address the dynamic nature of remote gate execution. For instance, remote gates executed earlier may trigger quantum information transfer that results in later local gates becoming remote. These limitations restrict the achievable performance of QAOA in DQC.

Therefore, a key challenge in DQC is to develop dedicated optimization techniques that fully account for the structural features of QAOA and the dynamic behavior of gates during execution, in order to minimize quantum communication overhead. Our experimental results indicate that the ordering of two-qubit gates in QAOA circuits has a noticeable impact on EPR pair usage. As shown in Fig. 1(b), the gap between the best and worst cases can reach up to 13.8%. Motivated

by this observation, we propose a set of optimization strategies to be applied during the circuit design stage. Building on the structural features of QAOA, we adopt a layer-wise circuit design approach, combined with iterative feedback from intermediate compilation results. Specifically, for each algorithmic layer, a partial circuit is constructed by carefully arranging its two-qubit gates. The resulting circuit is compiled once, and the compilation outcome is then used to guide the design of the next layer. This process captures the impact of inter-layer variations on subsequent execution. To further eliminate interference between local and remote two-qubit gates within the same layer, we handle them separately. This separation helps prevent additional EPR pair usage that may arise from quantum information transfer triggered by execution order.

In this work, we propose *HiQ-DF*, a QAOA circuit design framework tailored for DQC. *HiQ-DF* integrates a hierarchical optimization strategy operating on two complementary levels. (i) When constructing the partial circuit of each algorithmic layer, local and remote two-qubit gates are handled separately. Dedicated heuristic algorithms are then applied to primarily optimize circuit depth and EPR pair usage, respectively. (ii) In designing the complete QAOA circuit, a layer-wise iterative method is adopted, where the compilation feedback from each layer guides the construction of the next. This approach enables global optimization of the entire QAOA circuit.

The main contributions of this paper are as follows:

(I) We are the first, to the best of our knowledge, to experimentally demonstrate a key insight: optimizing the ordering of two-qubit gates within each algorithmic layer can effectively reduce quantum communication overhead. This reduction occurs during the execution of QAOA circuits on DQC systems.

(II) We propose *HiQ-DF*, the first QAOA circuit design framework tailored for DQC. By strategically optimizing the order of two-qubit gates derived from the cost Hamiltonian, *HiQ-DF* enables global circuit-level optimization.

(III) We introduce an innovative hierarchical optimization strategy in *HiQ-DF*, which consists of two main components. First, during the partial circuit design of each algorithmic layer, local and remote two-qubit gates are separately arranged using dedicated heuristic algorithms, resulting in two independent subcircuits. This separation allows for simultaneous optimization of both circuit depth and EPR pair usage. Second, when constructing the complete QAOA circuit, *HiQ-DF* employs a layer-wise iterative design method guided by compilation feedback. After completing the design of each layer, a round of circuit compilation is performed, and the resulting output is used to inform the design of the next layer. This process enables global optimization of the QAOA circuit.

(IV) Comprehensive experimental results show that QAOA circuits generated by *HiQ-DF* consistently outperform baseline approaches across three key metrics: EPR pair usage, circuit latency, and circuit depth. Specifically, *HiQ-DF* achieves an average reduction of 26.12% in EPR pair usage (up to 36.85%), 26.44% in circuit latency (up to 35.27%), and 39.63% in circuit depth (up to 49.3%).

The structure of this paper is as follows. Section 2 introduces QAOA for MaxCut, the implementation of QAOA circuits, and the fundamentals of DQC and quantum circuit compilation. Detailed design of our proposed *HiQ-DF* framework is presented in Section 3, followed by a thorough evaluation in Section 4. Related works are summarized in Section 5 and we finally conclude this paper in Section 6.

2. Background

2.1. QAOA for MaxCut and QAOA circuit

QAOA provides a hybrid quantum–classical approach for solving combinatorial optimization problems. A widely studied application is the MaxCut problem, which partitions the nodes of a graph into two sets to maximize the number of edges between them. Given a graph $G=(V,E)$ with $N=|V|$ nodes and $M=|E|$ edges, the corresponding cost function is defined as

$$C=\frac{1}{2}\sum_{(i,j)\in E}(1-\sigma_Z^i\sigma_Z^j), \tag{1}$$

where each edge connects nodes i and j, and σ_Z^i and σ_Z^j denote Pauli-Z operators applied to the qubits associated with these nodes.[8] A MaxCut solution can be represented by a bitstring state $|\psi\rangle$, where each qubit $s_i\in\{0,1\}$ indicates the partition assignment of node i: $s_i=0$ if node $i\in V_A$, and $s_i=1$ if node $i\in V_B$. The optimization goal is to maximize the expectation value of C with respect to the output state:

$$\langle C\rangle=\langle\psi(\gamma,\beta)|C|\psi(\gamma,\beta)\rangle. \tag{2}$$

QAOA begins by preparing the uniform superposition state $|+\rangle^{\otimes N}$, followed by p alternating layers of unitaries derived from the cost Hamiltonian H_{C} and the mixer Hamiltonian H_{M}. Each algorithmic layer applies a cost unitary $\mathrm{e}^{-\mathrm{i}\gamma_k H_{\mathrm{C}}}$ and a mixer unitary $\mathrm{e}^{-\mathrm{i}\beta_k H_{\mathrm{M}}}$. The resulting quantum state after p layers is given by

$$|\psi(\gamma,\beta)\rangle=\left(\prod_{k=0}^{p-1}\mathrm{e}^{-\mathrm{i}\beta_k H_{\mathrm{M}}}\,\mathrm{e}^{-\mathrm{i}\gamma_k H_{\mathrm{C}}}\right)|+\rangle^{\otimes N}, \tag{3}$$

where $\boldsymbol{\gamma}=(\gamma_0,\ldots,\gamma_{p-1})$ and $\boldsymbol{\beta}=(\beta_0,\ldots,\beta_{p-1})$ are classical parameters optimized by a classical outer loop. The total number of variational parameters is $2p$.

In QAOA circuits, the problem-specific cost Hamiltonian H_{C} and the mixer Hamiltonian H_{M} appear alternately. Once decomposed into their corresponding quantum gate sets, these Hamiltonians form the core structure of the circuit, as illustrated in Fig. 2. For example, in the MaxCut problem on an unweighted graph $G=(V,E)$, the cost Hamiltonian $H_{\mathrm{C}}=\sum_{(i,j)\in E}\frac{1}{2}\left(I-Z_iZ_j\right)$ is typically implemented using RZZ

(*ZZ*-rotation) gates that encode the interaction terms corresponding to the edges of the graph, while the mixer Hamiltonian $H_{\mathrm{M}} = \sum_{i \in V} X_i$ is implemented using *RX* (*X*-rotation) gates applied to all qubits to enable amplitude mixing.

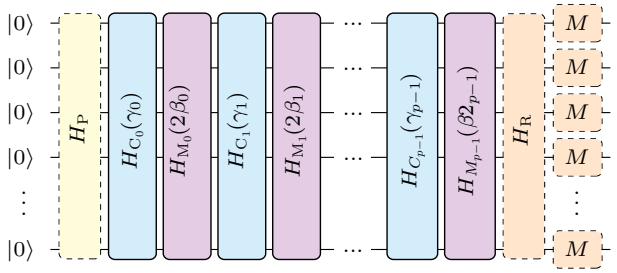


Fig. 2. The implementation of QAOA circuits.

If the initial state preparation and readout processes are regarded as part of parameter-free Hamiltonians,[42] then the implementation of QAOA circuits can be extended to four stages: H_{P}, H_{C}, H_{M}, and H_{R}. In this structure, H_{P} is the preparation Hamiltonian, which is typically implemented using Hadamard gates to create a uniform superposition state. H_{R} is the readout Hamiltonian, which performs basis transformation if measurements beyond the Z basis are needed. For different application scenarios, customized versions of H_{C} and H_{M} can be designed. With appropriate implementations of both state preparation and measurement, a complete QAOA circuit can thus be constructed to address a wide range of combinatorial optimization problems.

2.2. Distributed quantum computing

DQC offers a scalable paradigm to address computationally intensive problems by dividing quantum workloads across multiple quantum chips,[43,44] thereby enabling rapid expansion of quantum computational capabilities. In this subsection, we introduce two key infrastructures of DQC: remote quantum communication and quantum chip network.

The core of remote quantum communication is establishing remote EPR pairs (hereafter referred to simply as EPR pairs) between qubits on different chips, thereby enabling nonlocal quantum information transfer. Qubits used to form EPR pairs are referred to as *communication qubits*, and those for computation are *data qubits*. Currently, two main communication patterns are used in DQC: *Cat-Comm* and *TP-Comm*.[24] *Cat-Comm*, built with cat-entangler and cat-disentangler,[45] performs a remote CNOT between adjacent chips with one EPR pair. In contrast, *TP-Comm*, based on quantum teleportation,[46] requires two EPR pairs to perform a general remote two-qubit gate between adjacent chips. Therefore, the communication overhead of executing remote quantum gates depends not only on the chosen communication pattern but also on the physical distance between the involved chips.

A quantum chip network consists of multiple independent quantum chips interconnected via specially designed physical connections, such as superconducting coaxial cables.[16] These inter-chip connections enable the generation of entangled states between qubits residing on different chips, which in turn serve as quantum communication channels for transmitting quantum information across the network. Formally, a quantum chip network can be modeled as a four-tuple $\mathcal{N} = (C, \mathcal{Q}, E_{\mathrm{l}}, E_{\mathrm{r}})$, where (i) $C = (c_0, \ldots, c_{K-1})$ denotes the set of quantum chips, with K representing the total number of chips; (ii) $\mathcal{Q} = (\mathcal{Q}_0, \ldots, \mathcal{Q}_{K-1})$ is a list of physical qubit sets, where each $\mathcal{Q}_i$ corresponds to the qubits hosted on chip $c_i \in C$; (iii) E_{l} represents the set of intra-chip (local) qubit connections, where each $e_{\mathrm{l}} \in E_{\mathrm{l}}$ represents a pair of indices corresponding to the data qubits connected by that local physical link; (iv) E_{r} represents the set of inter-chip (remote) connections. Each $e_{\mathrm{r}} \in E_{\mathrm{r}}$ denotes a pair of data qubit indices, where each data qubit is connected to a communication qubit located at one end of the corresponding remote physical connection.

Based on this model, the connectivity between quantum chips is represented by a chip-level graph $G_{\mathrm{C}} = (C, E_{\mathrm{r}})$. Furthermore, by treating all physical qubits as nodes and including both local and remote couplings as edges, a global qubit connectivity graph can be defined as $G_Q = (Q, E_{\mathrm{p}})$, where $Q = \bigcup_{i=0}^{K-1} \mathcal{Q}_i$ is the set of all physical qubits, and $E_{\mathrm{p}} = E_{\mathrm{l}} \cup E_{\mathrm{r}}$ represents the union of local and remote connections.

2.3. Quantum circuit compilation

In NISQ devices, high-level quantum circuits cannot be executed directly due to hardware constraints such as limited topology connectivity and native gate sets. For instance, two-qubit gates work only between physically adjacent qubits, requiring SWAP gates to enable interactions between non-adjacent ones. To overcome these limitations, various quantum compilation techniques have been developed,[47–51] including gate decomposition, qubit mapping and routing, and basis gate translation. Among these, qubit mapping and qubit routing are two critical steps. Qubit mapping focuses on determining an optimal initial placement of logical qubits onto physical ones. In single-chip scenarios, the common objective is to minimize the number of additional SWAP gates required to execute all two-qubit operations.[50] Qubit routing, on the other hand, involves dynamically inserting SWAP gates during circuit execution to relocate logical qubits that are initially mapped to non-adjacent physical locations. This ensures that all quantum gates satisfy the physical topology constraints.[49]

In DQC, quantum circuit compilation is considerably more complex than in single-chip systems. This is not only due to limited intra-chip connectivity, but also stems from challenges such as high quantum communication overhead, non-fully-connected inter-chip connections, and the scarcity of communication resources like EPR pairs. To tackle these issues, numerous studies have proposed compilation

techniques,[24,39–41,43,44,52] with most efforts focused on minimizing the overhead associated with remote quantum communication. However, an efficient DQC compilation solution must also incorporate intra-chip compilation into a unified framework for co-optimization. For example, it requires end-to-end qubit mapping and routing at the physical qubit level, while jointly optimizing various costs arising in the distributed environment to enhance the overall execution efficiency of quantum circuits.

3. Hierarchical QAOA circuit design

3.1. Overview of *HiQ-DF*

The performance of QAOA in solving various problems on QCNs is significantly affected by remote communication overhead. To address this challenge, the *HiQ-DF* framework aims to minimize EPR pair usage during circuit execution by strategically ordering two-qubit gates in each algorithmic layer, while also optimizing overall circuit depth. To this end, it adopts a hierarchical optimization strategy with two levels. First, during the partial circuit design of each algorithmic layer, local and remote two-qubit gates are treated separately using dedicated heuristic algorithms. Second, in constructing the complete QAOA circuit, a layer-wise iterative design method is employed, where the compilation result of each layer provides feedback to guide the generation of the next. This hierarchical approach enables the global optimization of QAOA circuit design.

The overall structure of *HiQ-DF* framework is shown in Fig. 3. It consists of four core stages: stage I — preprocessing (Subsection 3.2), stage II — local subcircuit design (Subsection 3.3.1), stage III — remote subcircuit design (Subsection 3.3.2), and stage IV — qubit routing of partial circuit (Subsection 3.4). In addition, a simple example is presented in Subsection 3.5 to illustrate the workflow of *HiQ-DF*. Finally, Subsection 3.6 provides a detailed analysis of its time and space complexity.

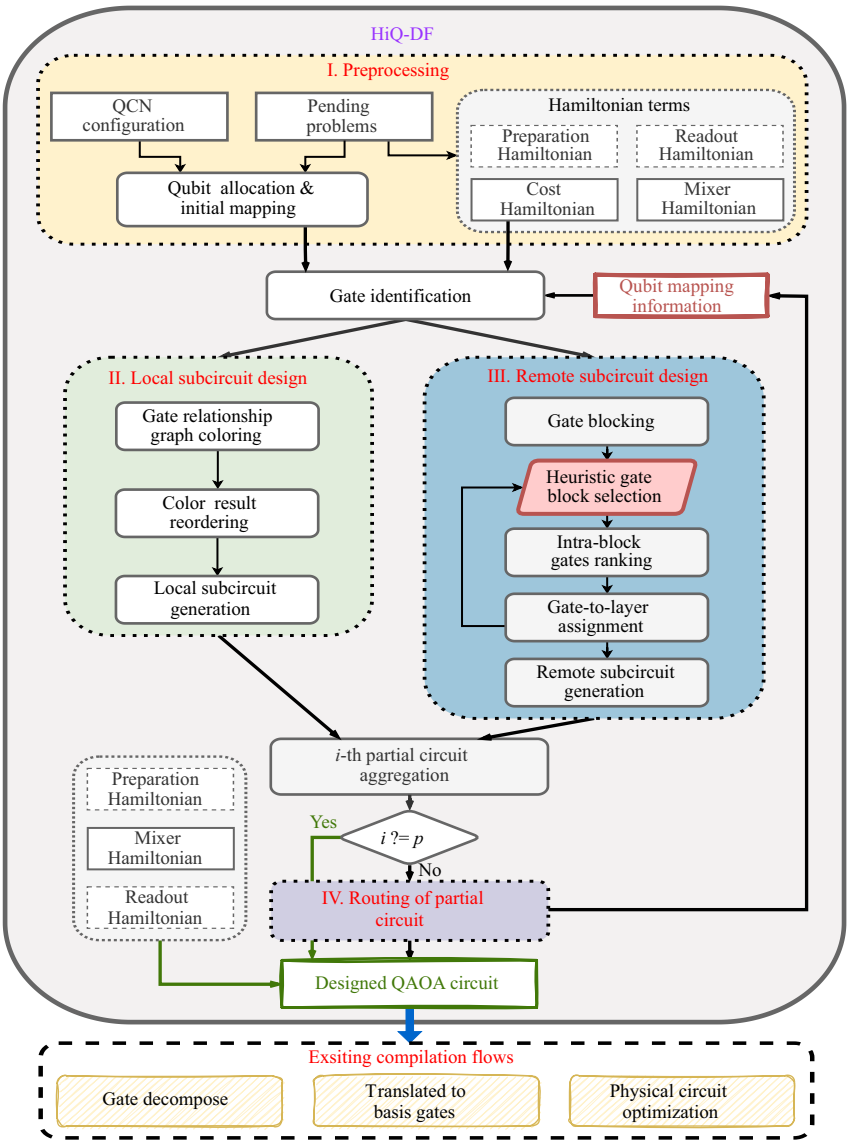


Fig. 3. Workflow of *HiQ-DF*.

3.2. Preprocessing

The primary objective of the preprocessing is to extract the set of two-qubit gates from the cost Hamiltonian and to generate an initial mapping from logical qubits to physical ones, which will be used for subsequent qubit routing. This stage takes as input both the problem definition (*e.g.*, a graph structure for MaxCut) and the configuration of the QCN, including the chip-level connectivity graph $G_{\mathrm{C}} = (C, E_{\mathrm{r}})$ and the global qubit connectivity graph $G_Q = (Q, E_{\mathrm{p}})$ (as described in Subsection 2.2). It consists of two modules: the Hamiltonian terms generation and the qubit allocation and initial mapping. The outputs of this stage collectively provide the structural basis for the circuit design process that follows.

(i) Hamiltonian terms generation This module is designed to extract the quantum gate sets corresponding to the Hamiltonians H_{P}, H_{C}, H_{M}, and H_{R}, which are defined in the four stages of QAOA circuit implementation as described in Subsection 2.1. The associated quantum gate sets are denoted as $\mathcal{G}_{H_{\mathrm{P}}}$, $\mathcal{G}_{H_{\mathrm{C}}}$, $\mathcal{G}_{H_{\mathrm{M}}}$, and $\mathcal{G}_{H_{\mathrm{R}}}$, respectively.

(ii) Qubit allocation and initial mapping This component performs the initial end-to-end placement of logical qubits onto physical ones in the QCN. This mapping plays a critical role in the subsequent circuit design process, as it enables the classification of the set of two-qubit gates $\mathcal{G}_{H_{\mathrm{C}}}$ into a local gate set $\mathcal{G}_{H_{\mathrm{C}}}^{\mathrm{loacal}}$ and a remote gate set $\mathcal{G}_{H_{\mathrm{C}}}^{\mathrm{remote}}$. Furthermore, after each execution of the qubit routing (step IV in Subsection 3.4), the mapping is updated to guide the next iteration of circuit design from step II to step IV (as shown in Fig. 3).

Several studies[39,40] have considered practical constraints in QCNs, such as limited chip capacity and restricted remote connections between quantum chips, and have proposed corresponding qubit allocation algorithms. These algorithms can be seamlessly integrated into the *HiQ-DF* framework. In our implementation, we adopt the MHSA algorithm[39] for qubit allocation. Since MHSA omits intra-chip qubit mapping, we introduce a basic strategy to achieve end-to-end initial qubit mapping. This strategy aims to assign frequently used logical qubits to highly connected physical qubits. Specifically, logical qubits are first sorted in descending order based on the frequency of two-qubit gates acting on them, while physical qubits are sorted according to their number of physical couplings. The most frequently used logical qubit is then mapped to the most highly connected physical qubit, and the remaining qubits are mapped one-to-one in order, ultimately producing the initial qubit placement.

3.3. LR subcircuit placement and partial circuit generation

Based on the qubit placement, the two-qubit gate set in $\mathcal{G}_{H_{\mathrm{C}}}$ can be divided into a local gate set, $\mathcal{G}_{H_{\mathrm{C}}}^{\mathrm{loacal}}$, and a remote gate set, $\mathcal{G}_{H_{\mathrm{C}}}^{\mathrm{remote}}$. Two-qubit gates in $\mathcal{G}_{H_{\mathrm{C}}}^{\mathrm{loacal}}$ act only on qubits within the same chip, while gates in $\mathcal{G}_{H_{\mathrm{C}}}^{\mathrm{remote}}$ involve operations across different chips. Using these gate sets, we can separately generate the local subcircuit (stage II) and the remote subcircuit (stage III). By integrating these two subcircuits, the partial QAOA circuit implementing the cost Hamiltonian in the i-th algorithmic layer (of p total layers) can be constructed. This design process can be further formulated as three hierarchical subproblems: (i) how to arrange the local two-qubit gates to construct the local subcircuit, (ii) how to arrange the remote two-qubit gates to construct the remote subcircuit, and (iii) how to determine the relative placement order of the local and remote subcircuits.

For the placement order, common strategies include local-then-remote (LR), remote-then-local (RL), and local-remote-local (LRL) sequences. Experimental results show that LR effectively reduces EPR pair overhead (more details provided in Subsection 4.3). This is because executing all local two-qubit gates first helps avoid scenarios where remote gates cause qubits to migrate between chips, which would otherwise turn local gates into remote ones. To enforce the execution of local gates before remote gates, a Barrier gate is inserted between local and remote subcircuits. Construction details for both subcircuits are given in the following sections.

3.3.1. Local subcircuit design

This stage aims to generate a local subcircuit with minimal depth by optimally arranging local two-qubit gates. As illustrated in Fig. 4, when multiple local two-qubit gates do not share any common qubits, they can be placed within the same gate layer and executed in parallel. Consequently, minimizing circuit depth is equivalent to maximizing the number of local two-qubit gates that can be assigned to the same gate layer. To this end, we reformulate the task of local subcircuit generation as the graph coloring problem. In this graph $G_{\mathrm{local}} = (V_{\mathrm{local}}, E_{\mathrm{local}})$, each node in V_{local} represents a local two-qubit gate, and an edge exists between two nodes in E_{local} if and only if the corresponding gates share at least one common qubit. Therefore, the minimum number of colors needed equals the minimum number of gate layers, i.e., the optimal circuit depth.

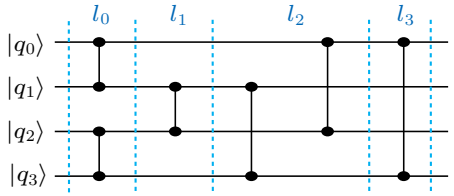


Fig. 4. An example of layered quantum circuit.

As illustrated in Fig. 3, this stage is structured into three sequential steps: gate relationship graph coloring, color result reordering, and local subcircuit generation. We begin by

applying the DSATUR algorithm[53] to the gate relationship graph G_{local} to obtain an optimized coloring solution, which effectively groups the local two-qubit gates into independent sets. These groups indicate gates that can be executed in parallel without conflicts. In the next step, the resulting groups are reordered in descending order based on the number of local two-qubit gates contained in each group, ensuring that denser gate sets are prioritized. Finally, each reordered group is assigned to a gate layer in ascending order, thereby constructing a local subcircuit with minimized depth and maximized gate parallelism.

3.3.2. Remote subcircuit design

In generating the remote subcircuit, our main goal is to minimize subsequent EPR pair usage through effective gate grouping and arrangement strategies, while reducing subcircuit depth. The process comprises five main steps, as shown in Fig. 3 and Algorithm 1.

Algorithm 1: Remote subcircuit design algorithm

Input: Remote two-qubit gates set $\mathcal{G}_{H_C}^{\text{remote}}$.

Output: Remote subcircuit $\mathcal{R}_\mathcal{SC}$.

1 Divide remote two-qubit gates from $\mathcal{G}_{H_C}^{\text{remote}}$ into non-overlapping blocks $\mathcal{B}_list = [\mathcal{B}_0, \ldots, \mathcal{B}_{K-1}]$;
2 Initialize empty gate layer set $\mathcal{L}$ and let N_{r} denote the number of remote two-qubit gates;
3 **while** $\mathcal{B}_list$ *is not none* **do**
4 $candidate_\mathcal{B}_list = Obtain_Candidate(\mathcal{B}_list, \mathcal{L})$;
5 **if** *len(candidate_B_list) > 1* **then**
6 Score = {};
7 **for** *block* $\mathcal{B}$ *in candidate_B_list* **do**
8 $\mathcal{L}' = \mathcal{L}.update(\mathcal{B})$;
9 $\mathcal{S} = Obtain_Candidate(\mathcal{B}_list, \mathcal{L}')$;
10 Score$[\mathcal{B}] = F(\mathcal{B}, N_{\text{r}}, \mathcal{L}, \mathcal{S}, \mathcal{L}')$;
11 **end**
12 Select block $\mathcal{B}_{\text{opt}}$ with the lowest cost score;
13 **end**
14 **else**
15 $\mathcal{B}_{\text{opt}} = candidate_\mathcal{B}_list[0]$;
16 **end**
17 Sort the gates in $\mathcal{B}_{\text{opt}}$ by metric $\mathcal{M}_2$ in descending order to obtain $sorted_\mathcal{B}_{\text{opt}}$;
18 **for** g *in* $sorted_\mathcal{B}_{\text{opt}}$ **do**
19 Find the earliest available layer in $\mathcal{L}$ for gate g and then update $\mathcal{L}$;
20 **if** *no such layer exists* **then**
21 Create a new layer in $\mathcal{L}$ and place g into it;
22 **end**
23 **end**
24 $\mathcal{B}$_list.$remove(\mathcal{B}_{\text{opt}})$;
25 **end**
26 Aggregate the remote two-qubit gates in each gate layer to generate the remote subcircuit $\mathcal{R}_\mathcal{SC}$;

From the overall workflow perspective, we first perform gate blocking (step 1) to obtain several two-qubit gate blocks. Then, steps 2 to 4 are executed in an iterative manner: in each iteration, one unassigned gate block is heuristically selected from all remaining blocks, followed by intra-block gate ranking and gate-to-layer assignment, until all gate blocks have been processed and assigned. Finally, the process enters the remote subcircuit generation (step 5), thus completing the construction of the subcircuit. The detailed design of each step is detailed below.

Step 1 Gate blocking The goal of gate blocking is to divide all remote two-qubit gates into several non-overlapping blocks (as shown in the line 1 of Algorithm 1). A block refers to a set of two-qubit gates that act on the same qubit, and non-overlapping means each gate is assigned to exactly one block without duplication.

To facilitate subsequent optimization, we group as many gates as possible into the same block using a greedy blocking strategy. Specifically, all qubits are first sorted in descending order according to the number of associated two-qubit gates. Then, we iterate through each qubit in the sorted list and assign all unallocated two-qubit gates associated with that qubit to the same block. Once all qubits have been traversed, all gates will have been assigned to their respective blocks, thus completing the blocking process. In addition, an empty list of gate layers $\mathcal{L}$ is initialized to store the two-qubit gates assigned to each block (as shown in the line 2 of Algorithm 1).

Step 2 Heuristic gate block selection During each iteration of block selection, the *Obtian_Candidate()* is first invoked to identify all candidate blocks from the set of unprocessed blocks (according to line 4 of Algorithm 1). These candidate blocks share a common characteristic: at least one two-qubit gate in the block can be placed into one of the current gate layers $\mathcal{L}$. If no such candidate exists, *Obtain_Candidate()* selects the highest-scoring block $\mathcal{B}$ from the list $\mathcal{B}_list$ based on the following metric:

$$\mathcal{M}_1 = \frac{\sum_{g_i \in \mathcal{B}} d_i}{len(\mathcal{B})}, \tag{4}$$

where d_i indicates the shortest distance (i.e., the remote communication distance) between the quantum chips to which the two qubits of the gate g_i are mapped.

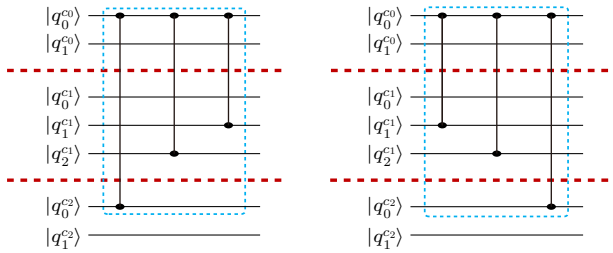
Otherwise, when candidate blocks are available, we apply a heuristic strategy to select the most appropriate block for placement. This strategy jointly considers the immediate EPR pair usage and the potential change in circuit depth resulting from placing the current block. In each iteration, the block with the lowest estimated cost is selected for subsequent processing (according to lines 5–13 of Algorithm 1). The cost function used in this heuristic approach is designed as follows:

$$\begin{aligned} & F(\mathcal{B}, N_{\mathrm{r}}, \mathcal{L}, \mathcal{S}, \mathcal{L}') \\ = \; & \omega_1 \cdot \sum_{g_i \in \mathcal{B}} \frac{N_{\mathrm{r}} - \mathcal{L}.g_i}{N_{\mathrm{r}}} \cdot d_i \\ & + \frac{\omega_2}{|\mathcal{S}|} \cdot \sum_{\mathcal{B}' \in \mathcal{S}} \left(\sum_{g_j \in \mathcal{B}'} \frac{N_{\mathrm{r}} - \mathcal{L}'.g_j}{N_{\mathrm{r}}} \cdot d_j \right). \end{aligned} \tag{5}$$

In Eq. (5), the cost function has two components weighted by tunable parameters ω_1 and ω_2. In the first, N_{r} denotes the total number of remote two-qubit gates, $\mathcal{L}.g_i$ is the index of the earliest possible layer in which the i-th gate in the current block can be placed within the current layer structure $\mathcal{L}$, and d_i indicates the remote communication distance between the quantum chips to which the two qubits of the i-th gate are mapped. In the second, $\mathcal{S}$ refers to the set of candidate lookahead blocks assuming the current block $\mathcal{B}$ has been placed in $\mathcal{L}$ and the layer structure is updated to $\mathcal{L}'$ (as shown in the lines 8–9 of Algorithm 1). Other symbols follow the definitions in the first component.

Step 3 Intra-block ranking After selecting a gate block, all two-qubit gates within it are sorted to determine processing order. Our observation reveals that prioritizing the execution of remote two-qubit gates with larger communication distances can help reduce the overall EPR pair consumption of the gate block.

As shown in Fig. 5, the two subfigures contain the same remote gates arranged in different orders. In Fig. 5(a), if the gate with communication distance $d = 2$ is executed first, it can be transformed into a $d = 1$ gate via the *TP-Comm* pattern (2 EPR pair usage) and then completed using the *Cat-Comm* pattern, consuming only 1-EPR pair. The remaining two gates become local due to the proximity of the involved qubits and require no EPR pairs, resulting in a total usage of 3 EPR pairs. In contrast, in Fig. 5(b), if the two $d = 1$ gates are executed first using the *Cat-Comm* pattern (incurring an optimal usage of 1 EPR pair), the subsequent $d = 2$ gate requires 3 additional EPR pairs, resulting in a total of 4 EPR pairs for the block.



(a) The first RZZ block example (b) The second RZZ block example

Fig. 5. An example of two gate blocks (panels (a) and (b)) that together contain three two-qubit gates acting on the same logical qubit q_0^{c0}.

This comparison demonstrates that prioritizing gates with higher communication overhead can reduce EPR pair usage. Motivated by this observation, we propose a new sorting metric and arrange all gates within the block in descending order according to this metric. The metric is defined as follows:

$$\mathcal{M}_2 = \frac{\deg(g.q_1) + \deg(g.q_2)}{2} \cdot d, \tag{6}$$

where $\deg(q)$ denotes the number of two-qubit gates associated with the qubit q (on which gate g acts) across all unscheduled blocks, $g.q_1$ and $g.q_2$ refer to the two qubits on which gate g acts, and d represents the communication distance of gate g.

Step 4 Gate-to-layer assignment For the sorted two-qubit gates within each block, a greedy strategy assigns them to the shallowest possible layers to minimize the overall circuit depth (as shown in lines 18–23 of Algorithm 1). The process begins with the first selected block. Since there are no existing layers at this point, we create as many new layers as there are gates in the block. Each gate is assigned to a separate layer according to the sorting order. For each subsequent block, each gate is placed in the earliest available layer where it does not conflict with existing gates. Two gates can only be placed in

the same layer if they operate on disjoint sets of qubits. If a gate cannot be assigned to any existing layer, a new layer is created for it. This approach ensures that each gate is allocated to a conflict-free layer.

Step 5 Remote subcircuit generation After all gate blocks have been processed, the final step is to aggregate the remote two-qubit gates assigned to each layer and generate the remote subcircuit. This subcircuit structure not only ensures efficient utilization of EPR pairs, but also minimizes the circuit depth. The resulting remote subcircuit can then be seamlessly integrated with the local subcircuits to form the partial circuit of QAOA.

3.4. Qubit routing of partial circuit

In QAOA circuits, each layer of the cost Hamiltonian corresponds to a partial circuit. During execution, the sets of local and remote two-qubit gates in these partial circuits may vary across different algorithmic layers. This variation arises from the movement of qubits after executing the previous layer, which alters the classification of gates in subsequent layers. Specifically, once a layer is executed, the updated qubit mapping may cause some gates that were previously local to become remote, and vice versa. These changes in gate classification directly affect the construction and optimization strategies of local and remote subcircuits in the following layer. To handle this, the *HiQ-DF* framework adopts a layer-wise iterative design method. After completing the partial circuit design, a qubit routing step is immediately performed to update the qubit mapping. The updated mapping is then used to reclassify two-qubit gates in the next algorithmic layer and guide the construction of the corresponding subcircuits. This iterative process enables cross-layer optimization and enhances the overall quality of QAOA circuit design.

To enable updates of the qubit mapping, each partial QAOA circuit must undergo end-to-end qubit routing at the physical qubit level. However, to the best of our knowledge, there is currently no qubit routing algorithm tailored for DQC that supports this capability. To address this limitation, we extend the widely adopted SABRE algorithm,[49] which was originally designed for single quantum chip, by modifying its heuristic SWAP insertion approach. The extended version, referred to as DiSAR, introduces a customized cost function that differentiates between the latency of executing two-qubit gates across chips (remote) and within the same chip (local), thereby better reflecting the differences in physical connectivity between and within quantum chips. The details of this cost function are provided below:

$$\begin{aligned} H = & \sum_{g\in F} \boldsymbol{D}[\pi(g.q_1)][\pi(g.q_2)] \\ & + \omega_{\mathrm{r}} \cdot \sum_{g\in F} f\left(\boldsymbol{D}'[\sigma(\pi(g.q_1))][\sigma(\pi(g.q_2))]\right) \\ & + \omega_3 \cdot \Bigg(\sum_{g\in E} \boldsymbol{D}[\pi(g.q_1)][\pi(g.q_2)] \\ & + \omega_{\mathrm{r}} \cdot \sum_{g\in E} f(\boldsymbol{D}'[\sigma(\pi(g.q_1))][\sigma(\pi(g.q_2))]) \Bigg). \end{aligned} \tag{7}$$

In Eq. (7), the parameter ω_{r} is the weight ratio between remote communication and local SWAP gate execution in terms of execution latency, while ω_3 (where $0 \le \omega_3 < 1$) reduces the influence of the associated extended set (F and E denote the front layer and the extended set, respectively, following the definition in SABRE[49]). The symbol π denotes the qubit mapping result, and σ represents the affiliation between the physical qubits and the quantum chip they are located in. The matrices $\boldsymbol{D}$ and $\boldsymbol{D}'$ encode the shortest distances between physical qubits and chips, respectively. In QCNs, both intra-chip and inter-chip physical connections are assigned a unit weight of 1. For chips connected via remote links, their inter-chip distance is also set to 1. Based on these assumptions, $\boldsymbol{D}$ and $\boldsymbol{D}'$ are computed using an all-pairs shortest path (APSP) algorithm, such as the Floyd–Warshall algorithm.[54] The function f is used to estimate the anticipated EPR pairs usage for executing remote two-qubit gates. Let c_i and c_j denote the quantum chips $\sigma(\pi(g.q_1))$ and $\sigma(\pi(g.q_2))$, respectively. The function $f(\boldsymbol{D}'[\sigma(\pi(g.q_1))][\sigma(\pi(g.q_2))])$ is defined as

$$f(\boldsymbol{D}'[c_i][c_j]) = 2 \cdot \boldsymbol{D}'[c_i][c_j] - 1, \tag{8}$$

where $\boldsymbol{D}'[c_i][c_j]$ denotes the communication distance between quantum chips c_i and c_j.

3.5. An example of QAOA circuit design

This section presents the complete workflow of *HiQ-DF* using a MaxCut problem as an example, solved by QAOA on a linearly connected QCN. As shown in Fig. 6(a), the input graph contains 12 vertices, where each vertex index corresponds to the index of a logical qubit in QAOA. Each edge is mapped to an RZZ gate acting on the associated qubit pair. As shown in Fig. 6(b), the QCN consists of three chips arranged linearly, each with equal capacity (4 physical qubits per chip). Next, we present a step-by-step illustration of how *HiQ-DF* designs the partial QAOA circuit corresponding to the Cost Hamiltonian within a single algorithmic layer.

(i) **Gates identification** After qubit allocation and initial mapping, the RZZ gates are classified into local and remote gate sets. The classification result is shown in Fig. 6(c).

(ii) **Local subcircuit design** Each local gate is represented as a vertex, and if two local gates act on the same qubit, an edge is added between their corresponding vertices, thereby constructing a conflict graph. A minimum graph coloring algorithm is then applied to this graph, producing two coloring groups. As shown in Fig. 6(d), the purple and green dashed boxes represent two sets of local gates with no qubit conflicts

within each set. Since the purple set contains more local gates than the green one, we assign them to gate layers $Layer_0$ and $Layer_1$, respectively. More details can be found in the corresponding part of Fig. 6(g).

(iii) **Remote subcircuit design** At this stage, the set of remote RZZ gates is first divided into four gate blocks through the gate blocking process, as illustrated in Fig. 6(e). These blocks are then prioritized using the sorting metric $\mathcal{M}_1$, resulting in an ordered list of blocks: $\mathcal{B}_list = [\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3]$. The detailed procedure of remote subcircuit construction is shown in Fig. 6(f). Initially, the gate layer set $\mathcal{L}$ is empty. Therefore, the algorithm selects the first block $\mathcal{B}_0$, sorts its internal RZZ gates based on metric $\mathcal{M}_2$, and creates new layers for each gate accordingly. The updated state of $\mathcal{L}$ is depicted in Fig. 6(f)(i). In the subsequent iterations, the algorithm chooses the next gate block from $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$. Since each of them contains at least one gate that can be placed in an existing layer of $\mathcal{L}$ without conflict, they are all considered candidates. Among them, $\mathcal{B}_1$ is selected as the next block to process because it contains a remote gate with a communication distance of 2, which implies a higher optimization priority. The result after processing block $\mathcal{B}_1$ is shown in Fig. 6(f)(ii). This process continues with blocks $\mathcal{B}_2$ (Fig. 6(f)(iii)) and $\mathcal{B}_3$ (Fig. 6(f)(iv)) in order, ultimately completing the construction of the remote subcircuit.

(iv) **Partial circuit aggregation** As shown in Fig. 6(f), the partial QAOA circuit is constructed by integrating gate sequences from local and remote subcircuits, with a Barrier gate inserted between them to enforce the execution of local gates before remote ones. The integrated circuit is then compiled using DiSAR, after which *HiQ-DF* proceeds to design the partial circuit for the next layer.

The *HiQ-DF* framework follows the above procedure to construct the partial QAOA circuit corresponding to the cost Hamiltonian in each algorithmic layer. After that, the gates associated with the mixer Hamiltonian are inserted accordingly. This process is repeated for all p layers. Finally, by adding the gates for the preparation and readout Hamiltonians at appropriate positions, the complete QAOA circuit design is finalized.

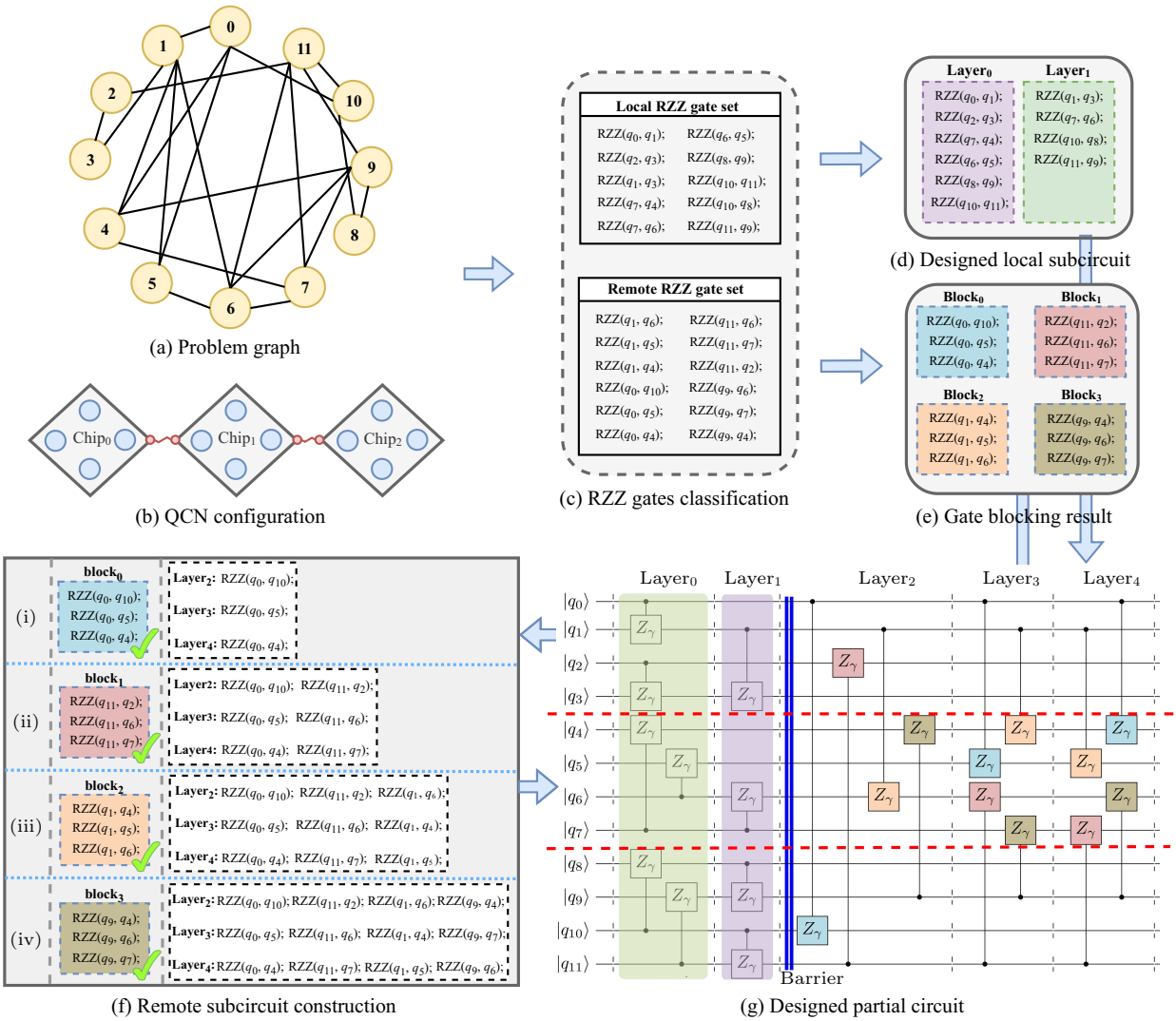


Fig. 6. A design example of partial QAOA circuit.

3.6. Time and space complexities analysis

In a typical QAOA setup, let N denote the number of logical qubits used. Each algorithmic layer contains M two-qubit gates derived from the cost Hamiltonian, and the complete QAOA circuit consists of p such layers. The target QCN comprises N_Q physical qubits. After performing gate classification, the total of M two-qubit gates is divided into two disjoint sets: M_1 local gates and M_2 remote gates, where $M = M_1 + M_2$.

The overall time complexity of the *HiQ-DF* framework consists of two main components: the preprocessing and p rounds of partial circuit design. In the preprocessing stage, the core task is qubit allocation and initial mapping. In the current implementation, this is performed using the MHSA algorithm, which incurs a time complexity of $O(N^4)$. During each round of circuit design, there are three key components: local subcircuit design with complexity $O(M_1^2)$, remote subcircuit design with complexity $O(M_2 \cdot N^3)$, and qubit routing with complexity $O(M \cdot N_Q^{2.5})$. Since remote subcircuit design dominates the computation, the overall time complexity of the circuit design stage is approximately $O(p \cdot M_2 \cdot N^3)$. Comparing both stages, the total time complexity of the *HiQ-DF* framework is $O(p \cdot M_2 \cdot N^3)$. As for space complexity, the primary contributor is the distance matrix that stores all pairwise distances between physical qubits. As this matrix is of size $N_Q \cdot N_Q$, the overall space complexity is $O(N_Q^2)$.

4. Evaluation

To evaluate the effectiveness of our proposed hierarchical QAOA circuit design framework, we perform experiments on a series of benchmarks derived from QAOA instances of the MaxCut problem. The source code implementation of the *HiQ-DF* framework and its comparison methods, as well as additional details on the benchmark programs used for evaluation, are available at https://github.com/RoccoLoter/HiQ-DF.

4.1. Experiment setup

Graph problems We evaluate our framework using MaxCut instances defined on two representative classes of graphs:

(i) *Regular graphs*: Each node has the same degree R (where R denotes the degree of each vertex), providing structured and homogeneous connectivity. We consider R ranging from 5 to 9, with the number of vertices (i.e., logical qubits) set between 50 and 150.

(ii) *Random graphs*: Generated according to the Erdös–Rényi model $G(N,P)$, where each of the $\binom{N}{2}$ possible edges is included independently with probability P. We vary P from 0.1 to 0.5, and set N in the same range of 50 to 150 vertices.

Quantum devices For our experimental evaluation, we choose the IBM quantum chip named *ibmq_guadalupe*[55] (device topology shown in Fig. 7(a)) to construct QCNs. Among these quantum chips, qubits Q_0, Q_6, Q_9, and Q_{15} are selected as candidate data qubits that are connected to communication qubits located on remote physical connections. In the six custom QCNs shown in Figs. 7(b)–7(g), each diamond-shaped block represents a chip, and each blue solid line between blocks denotes a remote physical connection between chips. For graph problems of different sizes, we conduct simulation experiments using corresponding QCN. For example, for a regular graph with $N = 50$ vertices, we use the $\text{QCN}_{4\text{chips}}$.

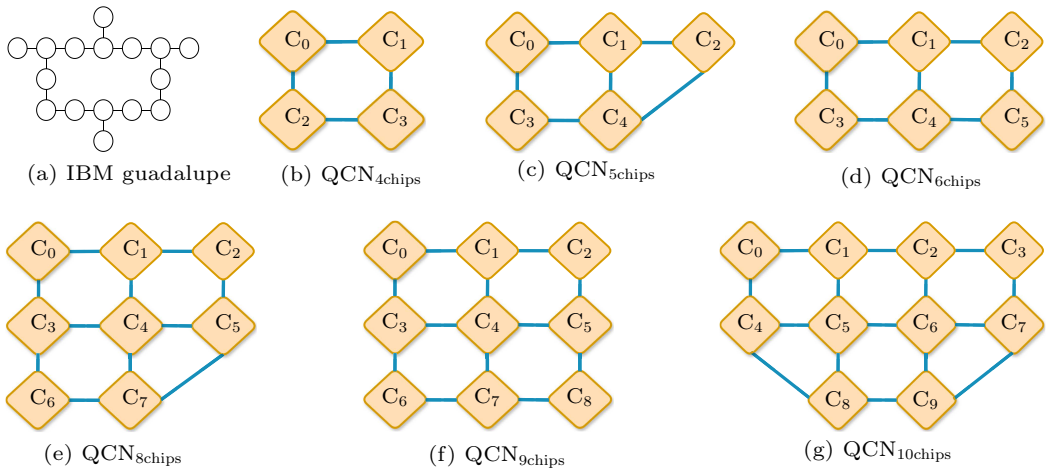


Fig. 7. The superconducting quantum chip used in the construction of QCNs, as well as QCNs with diverse architectures. (a) The device topologies of *ibmq_guadalupe* quantum chip. (b)–(g) QCNs constructed using 4, 5, 6, 8, 9, and 10 *ibmq_guadalupe* chips, respectively.

Metrics To evaluate our proposed framework, the following three metrics are used. (i) **EPR pairs usage** (lower is better): This metric indicates the estimated number of EPR pairs consumed during the execution of compiled QAOA circuits. We assume that executing a remote RZZ gate between adjacent quantum processors consumes one EPR pair (based on the *Cat-Comm* pattern), while executing a remote SWAP gate consumes two EPR pairs (based on the *TP-Comm* pattern). (ii) **Circuit latency** (lower is better): This metric reflects the estimated overall execution time overhead of the compiled

quantum circuit, calculated by summing the latencies of all quantum gates. Following the configuration,[56] we set the latencies of single-qubit gates and local SWAP gates to 1 and 6, respectively, and assign a latency of 5 to local RZZ gates. For remote quantum gates, according to the settings,[24] we set the latency of remote RZZ gates and remote SWAP gates executed on adjacent quantum processors to 60 and 120, respectively. (iii) **Compiled circuit depth** (lower is better): The depth of the compiled QAOA circuit.

Baseline methods To the best of our knowledge, there is currently no existing work specifically focused on designing QAOA circuits for DQC. Therefore, we propose four circuit design methods as baseline methods: *Total_Random*, *RL_Random*, *LR_Random*, and *LRL_Random*. In the *Total_Random* method, all RZZ gates are randomly ordered to generate the complete QAOA circuit. In the *RL_Random* and *LR_Random* methods, local and remote RZZ gates are randomly ordered separately to form corresponding subcircuits, which are then sequentially placed in a remote-then-local (RL) or local-then-remote (LR) manner to construct a partial QAOA circuit. In the *LRL_Random* method, the set of local RZZ gates is divided into two parts to create two local subcircuits, which are placed together with the remote subcircuit in a local-remote-local (LRL) order to form a partial circuit. Each of these partial circuits is repeated p times to construct the final complete QAOA circuit.

Parameters setting The parameters in Eqs. (5) and (7) are set to the following values throughout the evaluation: $\omega_{\mathrm{l}} = 1$, $\omega_2 = 0.5$, $\omega_3 = 0.5$, and $\omega_{\mathrm{r}} = 2$. A relatively small value is assigned to ω_{r} to avoid the risk of DiSAR algorithm entering cyclic traps during the routing process.

4.2. Experiment design

To enable the *HiQ-DF* framework to achieve better performance, we incorporate multiple design strategies. To evaluate the effectiveness of the framework and these individual strategies, we conduct a series of experiments and analyze the results based on the following key questions:

(i) Q1: How does the LR subcircuit placement strategy compare to other strategies?

(ii) Q2: How effective is the heuristic algorithm in *HiQ-DF* for remote subcircuit design?

(iii) Q3: How does the *HiQ-DF* perform overall in QAOA circuit design?

To address the first question, we evaluate the impact of different subcircuit placement strategies on EPR pair usage in QAOA circuit design. Specifically, we generate complete QAOA circuits ($p = 1$) for MaxCut problems on regular graphs ($R = 5$) and random graphs ($P = 0.5$) with sizes ranging from 50 to 150 vertices, using four methods: *Total_Random*, *RL_Random*, *LR_Random*, and *LRL_Random*. These methods represent four different subcircuit placement strategies. Among them, *RL_Random*, *LR_Random*, and *LRL_Random* use the same ordering of RZZ gates within the constructed remote subcircuits. Furthermore, *RL_Random* and *LR_Random* also share the same ordering in their local subcircuits. This setup allows for a fair comparison of placement strategies while keeping gate content consistent. After construction, all circuits are compiled using the DiSAR routing algorithm, and estimated EPR pair usage is recorded as the performance metric.

To address the second question, we design remote subcircuits using two strategies, while local subcircuits share the same random ordering. These subcircuits are then assembled using the LR placement strategy to construct QAOA circuits ($p = 1$) for the MaxCut problem on regular graphs ($R = 5$) and random graphs ($P = 0.5$) with sizes ranging from 50 to 150 vertices. All circuits are compiled using the DiSAR algorithm, and their performance is evaluated based on three metrics: EPR pair usage, circuit latency, and circuit depth. This enables a direct comparison between the heuristic remote subcircuit design in *HiQ-DF* and a purely random approach.

To address the third question, two QAOA circuit design methods, *Total_Random* and *LR_Random*, are selected as baselines for comparison with *HiQ-DF*. QAOA circuits ($p = 10$) are designed for the MaxCut problem on regular and random graphs of varying sizes, with the number of vertices ranging from 50 to 150. For regular graphs, the degree parameter R is set between 5 and 9, while for random graphs, the edge probability P ranges from 0.1 to 0.5.

4.3. The effectiveness of LR subcircuit placement strategy

To evaluate the impact of different subcircuit placement strategies on communication overhead, we compare the EPR pair usage of QAOA circuits generated by four representative design methods. These methods include *Total_Random*, *RL_Random*, *LR_Random*, and *LRL_Random* (more details are provided in Subsection 4.1).

Figure 8 presents the comparison of EPR pair usage after compilation for QAOA circuits ($p = 1$) generated by the four methods on regular graphs ($R = 5$) and random graphs ($P = 0.5$), with graph sizes ranging from 50 to 150 vertices. The x-axis denotes the number of vertices in the problem graph, while the y-axis shows the EPR pair usage details. For each method, the circuit generation is repeated ten times, and the average EPR pair overhead is calculated.

The results clearly show that *LR_Random* consistently achieves the lowest EPR pair usage across all graph sizes in both graph types. These findings indicate that the local-then-remote subcircuit placement strategy plays a positive role in reducing remote communication overhead during QAOA circuit execution.

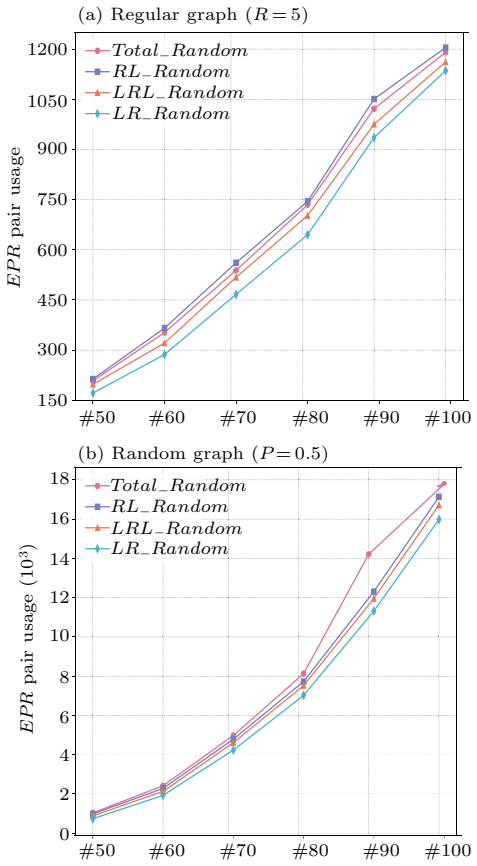


Fig. 8. For the MaxCut problem on (a) regular graphs ($R = 5$) and (b) random graphs ($P = 0.5$), the EPR pair usage of compiled QAOA circuits ($p = 1$) generated by four random circuit design methods is compared.

4.4. The effectiveness of heuristic remote subcircuit design algorithm

To evaluate the effectiveness of the heuristic remote subcircuit design algorithm proposed in the *HiQ-DF* framework, we conduct a comparative study against the baseline method *LR_Random*. The experiments are performed on both regular graphs ($R = 5$) and random graphs ($P = 0.5$), with graph sizes ranging from 50 to 150 vertices. For each graph instance, QAOA circuits ($p = 1$) are generated and compiled, and three key performance metrics are analyzed: EPR pair usage, compiled circuit latency, and circuit depth. As shown in Fig. 9, the comparison results are as follows:

(i) EPR pair usage (Figs. 9(a) and 9(d)): *HiQ-DF* significantly reduces the number of EPR pairs required for remote communication. For QAOA circuits generated for regular graphs, the EPR pair usage is reduced by approximately 23.35%, while for random graphs, the reduction is about 7.82%.

(ii) Compiled circuit latency (Figs. 9(b) and 9(e)): The total execution latency of the compiled circuits is also notably reduced. For QAOA circuits targeting regular graphs, *HiQ-DF* achieves a latency reduction of 24.36%. For random graphs, the reduction is 9.34%.

(iii) Circuit depth (Figs. 9(c) and 9(f)): *HiQ-DF* also reduces overall circuit depth, achieving 37.81% and 31.88% reductions for QAOA circuits on regular and random graphs, respectively.

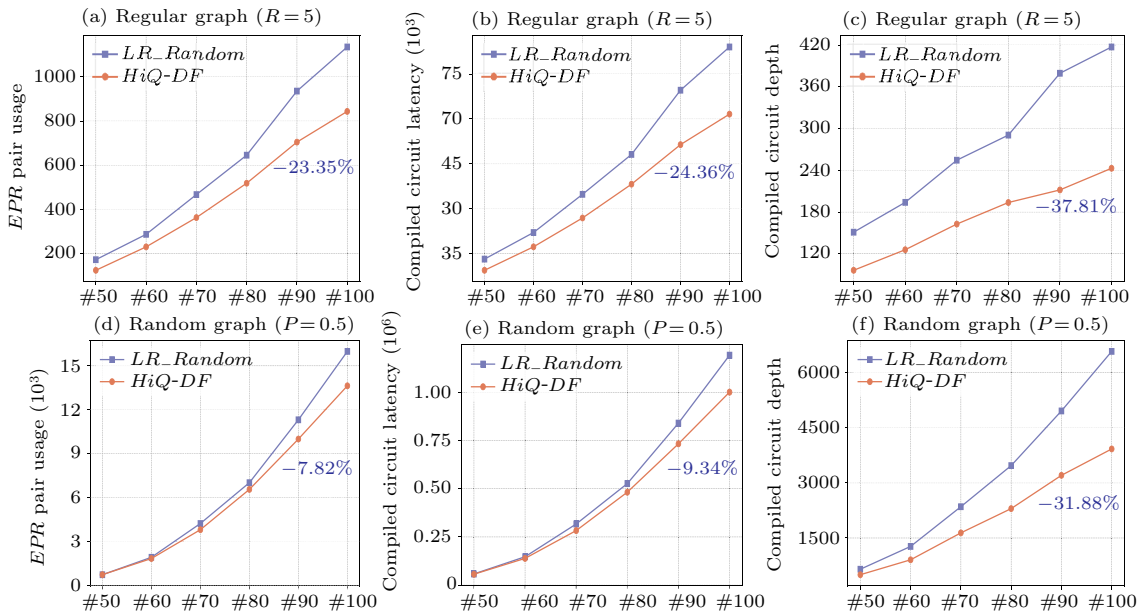


Fig. 9. For the MaxCut problem on regular graphs ($R = 5$) and random graphs ($P = 0.5$), the QAOA circuits ($p = 1$) generated by *HiQ-DF* and *LR_Random* are compiled and compared in terms of three key metrics: EPR pair usage ((a) and (d)), circuit latency ((b) and (e)), and circuit depth ((c) and (f)).

These results demonstrate that the heuristic remote subcircuit design algorithm in *HiQ-DF* consistently improves circuit performance across different graph structures and problem sizes. It not only effectively reduces EPR pair usage but also enhances the overall execution efficiency of QAOA circuits.

4.5. Overall comparison

To comprehensively evaluate the effectiveness of *HiQ-DF*, we compare it with two baseline methods, *Total_Random* and

LR_Random, by designing QAOA circuits ($p = 10$) on both regular and random graphs. The comparison is conducted using three key metrics: EPR pair usage, compiled circuit latency, and compiled circuit depth. The detailed results are presented in Tables 1 and 2.

Table 1. Comparison result of compiled QAOA circuits ($p = 10$) generated by different design methods for the MaxCut problem on regular graphs.

Vertices	R	*Total_Random*			*LR_Random*			*HiQ-DF*			*Total_Random* v.s. *HiQ-DF*			*LR_Random* v.s. *HiQ-DF*		
		EPR	*latency*	*depth*	*EPR*	*latency*	*depth*	*EPR*	*latency*	*depth*	ΔEPR	$\Delta latency$	$\Delta depth$	ΔEPR	$\Delta latency$	$\Delta depth$
50	5	1861.4	139650.0	1464.7	1863.2	139942.8	1515.7	1204.9	91698.4	953.7	**35.27%**	**34.34%**	**34.89%**	**35.33%**	**34.47%**	**37.08%**
	6	2408.7	181595.2	1941.6	2322.9	175320.4	1887.4	1611.9	121961.8	1256.5	**33.08%**	**32.84%**	**35.29%**	**30.61%**	**30.43%**	**33.43%**
	7	2954.9	221771.0	2295.7	2856.8	215000.6	2331.8	1905.3	144483.4	1445.7	**35.52%**	**34.85%**	**37.03%**	**33.31%**	**32.8%**	**38.0%**
	8	3305.6	248328.6	2543.2	3251.1	244453.2	2448.3	2122.8	161163.4	1515.4	**35.78%**	**35.1%**	**40.41%**	**34.71%**	**34.07%**	**38.1%**
	9	3778.0	282758.2	3013.5	3684.8	276921.4	2971.5	2431.3	184315.6	1817.6	**35.65%**	**34.82%**	**39.68%**	**34.02%**	**33.44%**	**38.83%**
70	5	3107.2	232786.2	1864.6	3029.7	226941.6	1962.1	2144.4	161262.2	1239.0	**30.99%**	**30.73%**	**33.55%**	**29.22%**	**28.94%**	**36.85%**
	6	4009.7	299979.8	2460.6	3900.4	291717.2	2443.5	2728.0	204901.4	1496.2	**31.96%**	**31.69%**	**39.19%**	**30.06%**	**29.76%**	**38.77%**
	7	4678.7	351443.2	2967.3	4588.4	345370.0	2941.6	3179.9	239360.0	1787.8	**32.03%**	**31.89%**	**39.75%**	**30.7%**	**30.69%**	**39.22%**
	8	5568.9	417059.4	3387.4	5437.6	408884.4	3443.6	3782.2	283846.4	2063.2	**32.08%**	**31.94%**	**39.09%**	**30.44%**	**30.58%**	**40.09%**
	9	6320.3	472275.2	3926.4	6199.8	463651.4	3882.3	4375.7	327097.4	2382.0	**30.77%**	**30.74%**	**39.33%**	**29.42%**	**29.45%**	**38.64%**
90	5	4751.6	352837.2	2437.1	4674.5	348018.0	2412.0	3356.8	248811.6	1525.5	**29.35%**	**29.48%**	**37.41%**	**28.19%**	**28.51%**	**36.75%**
	6	6021.9	447204.0	3081.9	5841.9	434568.6	3017.5	4329.9	320006.4	1881.9	**28.1%**	**28.44%**	**38.94%**	**25.88%**	**26.36%**	**37.63%**
	7	7445.1	553693.2	3923.6	7271.4	540594.0	3737.0	5257.3	388955.4	2319.3	**29.39%**	**29.75%**	**40.89%**	**27.7%**	**28.05%**	**37.94%**
	8	8674.9	644817.6	4537.7	8614.0	638807.4	4461.0	6239.0	459639.0	2658.4	**28.08%**	**28.72%**	**41.42%**	**27.57%**	**28.05%**	**40.41%**
	9	9815.2	728335.8	5122.1	9602.8	713577.0	5076.6	7189.7	530541.6	3130.0	**26.75%**	**27.16%**	**38.89%**	**25.13%**	**25.65%**	**38.34%**
110	5	6691.8	497206.8	3053.5	6608.4	488235.6	2882.2	4825.8	355476.4	1822.4	**27.88%**	**28.51%**	**40.32%**	**26.97%**	**27.19%**	**36.77%**
	6	8315.8	615108.4	3740.5	8278.8	611184.4	3632.7	6087.2	445619.2	2188.3	**26.8%**	**27.55%**	**41.5%**	**26.47%**	**27.09%**	**39.76%**
	7	9964.4	740700.2	4580.2	9798.7	727673.0	4504.5	7485.8	550150.6	2757.3	**24.87%**	**25.73%**	**39.8%**	**23.6%**	**24.4%**	**38.79%**
	8	11692.8	869151.0	5409.1	11596.5	861565.2	5373.0	8672.1	638345.8	3174.0	**25.83%**	**26.56%**	**41.32%**	**25.22%**	**25.91%**	**40.93%**
	9	13166.7	984001.0	6416.3	13044.4	974929.0	6274.3	10017.6	740416.0	3819.4	**23.92%**	**24.75%**	**40.47%**	**23.2%**	**24.05%**	**39.13%**
130	5	8858.1	654040.8	3559.5	8697.7	643083.0	3556.7	6535.9	478025.0	2138.1	**26.22%**	**26.91%**	**39.93%**	**24.85%**	**25.67%**	**39.89%**
	6	11081.4	819612.8	4448.6	11000.0	812459.6	4432.6	8426.4	614647.4	2750.7	**23.96%**	**25.01%**	**38.17%**	**23.4%**	**24.35%**	**37.94%**
	7	13618.5	1007507.2	5669.4	13553.5	999229.0	5379.5	11245.0	839454.8	3655.7	**17.43%**	**16.68%**	**35.52%**	**17.03%**	**15.99%**	**32.04%**
	8	15582.0	1153758.0	6577.6	15320.4	1133541.0	6357.8	11829.2	863413.4	3777.2	**24.08%**	**25.17%**	**42.57%**	**22.79%**	**23.83%**	**40.59%**
	9	17543.7	1302145.4	7564.0	17433.0	1290273.2	7136.8	13593.3	993600.2	4377.0	**22.52%**	**23.7%**	**42.13%**	**22.03%**	**22.99%**	**38.67%**
150	5	10421.7	774396.6	4033.9	10292.5	763320.6	3807.9	7774.7	569582.4	2336.4	**25.4%**	**26.45%**	**42.08%**	**24.46%**	**25.38%**	**38.64%**
	6	13082.4	974356.2	5071.2	12814.2	950988.0	4799.1	9990.9	731523.0	2986.4	**23.63%**	**24.92%**	**41.11%**	**22.03%**	**23.08%**	**37.77%**
	7	15962.8	1185890.4	6128.5	15753.3	1167807.6	5764.9	12347.4	903393.0	3603.0	**22.65%**	**23.82%**	**41.21%**	**21.62%**	**22.64%**	**37.5%**
	8	18002.5	1340547.6	7046.7	17880.7	1330892.4	6962.6	14062.8	1029684.6	4077.7	**21.88%**	**23.19%**	**42.13%**	**21.35%**	**22.63%**	**41.43%**
	9	21098.9	1569970.2	8426.1	20920.1	1555059.6	8115.3	16402.0	1202406.0	4773.3	**22.26%**	**23.41%**	**43.35%**	**21.6%**	**22.68%**	**41.18%**
Average Improvement											**27.8%**	**28.16%**	**39.58%**	**26.63%**	**26.97%**	**38.37%**

ΔEPR, $\Delta latency$, and $\Delta depth$ denote the percentage reductions in EPR pair usage, overall circuit latency, and circuit depth, respectively. For the comparison between *LR_Random* and *HiQ-DF*: $\Delta EPR = 1 - EPR(HiQ\text{-}DF)/EPR(LR_Random)$, $\Delta latency = 1 - latency(HiQ\text{-}DF)/latency(LR_Random)$, $\Delta depth = 1 - depth(HiQ\text{-}DF)/depth(LR_Random)$.

Table 2. Comparison result of compiled QAOA circuits ($p = 10$) generated by different design methods for the MaxCut problem on random graphs.

Vertices	P	*Total_Random*			*LR_Random*			*HiQ-DF*			*Total_Random* v.s. *HiQ-DF*			*LR_Random* v.s. *HiQ-DF*		
		EPR	*latency*	*depth*	*EPR*	*latency*	depth	*EPR*	*latency*	*depth*	ΔEPR	$\Delta latency$	$\Delta depth$	ΔEPR	$\Delta latency$	$\Delta depth$
50	0.5	2027.2	151047.6	1513.5	1947.2	146188.8	1560.4	1280.2	97780.0	1002.3	**36.85%**	**35.27%**	**33.78%**	**34.25%**	**33.11%**	**35.77%**
	0.6	4368.2	326492.0	3433.3	4227.3	316541.0	3345.0	2865.9	217116.4	2062.1	**34.39%**	**33.5%**	**39.94%**	**32.2%**	**31.41%**	**38.35%**
	0.7	6081.3	451687.4	4805.8	5814.2	433548.2	4591.1	4015.6	302569.6	2954.6	**33.97%**	**33.01%**	**38.52%**	**30.93%**	**30.21%**	**35.65%**
	0.8	8526.5	636084.0	6466.7	7950.5	593776.8	6127.8	5610.6	425789.8	3891.4	**34.2%**	**33.06%**	**39.82%**	**29.43%**	**28.29%**	**36.5%**
	0.9	10570.9	788733.8	8093.2	9554.5	714438.8	7477.4	6887.1	521723.8	4686.7	**34.85%**	**33.85%**	**42.09%**	**27.92%**	**26.97%**	**37.32%**
70	0.5	4739.2	352830.4	3109.6	4602.2	342867.4	2982.4	3301.6	246468.2	1894.8	**30.33%**	**30.15%**	**39.07%**	**28.26%**	**28.12%**	**36.47%**
	0.6	9437.1	703580.2	5892.4	9272.4	691403.2	5743.8	6715.5	500579.6	3583.2	**28.84%**	**28.85%**	**39.19%**	**27.58%**	**27.6%**	**37.62%**
	0.7	15003.7	1120084.0	9319.3	14541.2	1086237.4	9088.7	10796.3	806116.4	5539.7	**28.04%**	**28.03%**	**40.56%**	**25.75%**	**25.79%**	**39.05%**
	0.8	18804.1	1403139.8	11864.2	17824.1	1329525.8	11227.7	13443.5	1004646.2	6810.8	**28.51%**	**28.4%**	**42.59%**	**24.58%**	**24.44%**	**39.34%**
	0.9	24250.2	1807145.4	15114.2	22599.3	1685260.2	14143.5	17073.9	1276374.8	8530.8	**29.59%**	**29.37%**	**43.56%**	**24.45%**	**24.26%**	**39.68%**
90	0.5	9651.9	714263.4	5283.6	9497.7	703434.0	5236.4	7175.9	528758.4	3314.1	**25.65%**	**25.97%**	**37.28%**	**24.45%**	**24.83%**	**36.71%**
	0.6	19458.0	1438246.8	10527.9	18988.3	1404283.2	9985.5	14686.1	1083490.2	6373.0	**24.52%**	**24.67%**	**39.47%**	**22.66%**	**22.84%**	**36.18%**
	0.7	29153.8	2155542.4	15658.1	28132.3	2081262.4	15155.5	21931.0	1617113.4	9269.2	**24.77%**	**24.98%**	**40.8%**	**22.04%**	**22.3%**	**38.84%**
	0.8	38832.1	2868185.4	20756.9	37034.1	2738793.0	19967.5	28544.6	2107023.0	12181.1	**26.49%**	**26.54%**	**41.32%**	**22.92%**	**23.07%**	**39.0%**
	0.9	49820.6	3679929.6	26692.9	47424.7	3507584.4	25577.9	35129.9	2598225.0	15045.7	**29.49%**	**29.39%**	**43.63%**	**25.92%**	**25.93%**	**41.18%**
110	0.5	15400.9	1138827.2	7454.3	15228.8	1125456.2	7173.2	11865.0	867490.0	4392.4	**22.96%**	**23.83%**	**41.08%**	**22.09%**	**22.92%**	**38.77%**
	0.6	31691.0	2354420.2	15716.9	31073.2	2309075.8	15331.6	25017.0	1849146.4	9758.0	**21.06%**	**21.46%**	**37.91%**	**19.49%**	**19.92%**	**36.35%**
	0.7	48472.7	3583142.2	23024.4	47249.1	3496269.4	22412.4	37996.2	2785809.4	13630.3	**21.61%**	**22.25%**	**40.8%**	**19.58%**	**20.32%**	**39.18%**
	0.8	65196.9	4841510.0	32023.8	62573.4	4648520.6	30604.6	50069.6	3686780.2	18543.1	**23.2%**	**23.85%**	**42.1%**	**19.98%**	**20.69%**	**39.41%**
	0.9	82011.1	6065985.2	38872.8	77690.4	5743826.6	36955.0	59943.0	4400390.2	21191.1	**26.91%**	**27.46%**	**45.49%**	**22.84%**	**23.39%**	**42.66%**
130	0.5	24966.5	1845068.8	10953.7	24477.3	1809471.4	10753.0	19712.89	1445355.33	7008.0	**21.04%**	**21.66%**	**36.02%**	**19.46%**	**20.12%**	**34.83%**
	0.6	49280.9	3641239.8	21722.8	48453.7	3575401.8	20947.7	39767.33	2914276.67	13306.89	**19.3%**	**19.96%**	**38.74%**	**17.93%**	**18.49%**	**36.48%**
	0.7	75387.8	5564713.8	33080.6	73310.4	5413964.4	32032.1	59539.0	4363367.33	19580.78	**21.02%**	**21.59%**	**40.81%**	**18.79%**	**19.41%**	**38.87%**
	0.8	101729.6	7489510.8	43786.1	98530.1	7255819.8	42312.3	77618.0	5665610.0	24699.4	**23.7%**	**24.35%**	**43.59%**	**21.22%**	**21.92%**	**41.63%**
	0.9	127070.7	9366581.4	54505.4	120902.9	8917081.8	52222.1	92108.6	6732705.2	29533.2	**27.51%**	**28.12%**	**45.82%**	**23.82%**	**24.5%**	**43.45%**
150	0.5	34579.7	2568151.2	14473.1	34176.8	2538470.4	14097.8	28200.0	2066889.67	8558.44	**18.45%**	**19.52%**	**40.87%**	**17.49%**	**18.58%**	**39.29%**
	0.6	68178.6	5064384.6	28373.8	67211.9	4993521.6	27414.9	55657.1	4080096.6	15949.1	**18.37%**	**19.44%**	**43.79%**	**17.19%**	**18.29%**	**41.82%**
	0.7	105674.8	7844702.4	43293.5	103310.5	7670956.2	42790.4	84337.9	6183226.2	24105.0	**20.19%**	**21.18%**	**44.32%**	**18.36%**	**19.39%**	**43.67%**
	0.8	141281.0	10488553.8	58521.5	136625.9	10138297.2	56093.2	105463.7	7737251.4	30514.2	**25.35%**	**26.23%**	**47.86%**	**22.81%**	**23.68%**	**45.6%**
	0.9	177505.9	13162141.0	72494.0	169937.3	12593575.6	69111.3	124864.5	9164959.8	36751.8	**29.66%**	**30.37%**	**49.3%**	**26.52%**	**27.23%**	**46.82%**
Average Improvement											**26.36%**	**26.54%**	**41.34%**	**23.7%**	**23.93%**	**39.22%**

ΔEPR, $\Delta latency$, and $\Delta depth$ are computed in the same way as in Table 1.

(i) For regular graphs with degrees ranging from 5 to 9 (Table 1), *HiQ-DF* consistently outperforms the baselines across all three metrics. Compared with *Total_Random*, *HiQ-DF* achieves an average reduction of 27.8% in EPR pair usage, 28.16% in circuit latency, and 39.58% in circuit depth. Compared with *LR_Random*, the corresponding improvements are 26.63%, 26.97%, and 38.37%, respectively. These results demonstrate that *HiQ-DF* effectively reduces remote communication overhead and compresses execution latency and circuit depth, particularly on structured graph instances.

(ii) On random graphs with edge probabilities ranging from 0.5 to 0.9 (Table 2), *HiQ-DF* still shows consistent performance advantages. When compared with *Total_Random*, it reduces EPR pair usage, circuit latency, and depth by an average of 26.36%, 26.54%, and 41.34%, respectively. Against *LR_Random*, the average improvements are 23.7%, 23.93%, and 39.22%. Despite the irregular and sparse connectivity of random graphs, *HiQ-DF* maintains strong optimization capability, indicating the generality and robustness of its remote subcircuit design and placement strategy.

In summary, *HiQ-DF* outperforms baseline methods across various graph structures and problem sizes, demonstrating its scalability and practical effectiveness in large-scale QAOA circuit design.

5. Conclusion and future work

The circuit implementation of QAOA plays a critical role in determining its problem-solving performance on real quantum devices, especially in DQC scenarios. Through experimental observations, we identified that the execution order of two-qubit gates can significantly affect the remote communication overhead during circuit execution. To address this, we proposed *HiQ-DF*, a QAOA circuit design framework tailored for DQC, which hierarchically optimizes the ordering of different types of two-qubit gates within each algorithmic layer. This hierarchical optimization not only reduces potential remote communication overhead but also compresses the overall circuit depth, thereby improving circuit execution efficiency. Experimental results demonstrate that QAOA circuits generated by *HiQ-DF*, after compilation, consistently outperform those produced by baseline methods in key metrics, including EPR pair usage, circuit latency, and circuit depth, thereby confirming its significant performance gains.

Moreover, *HiQ-DF* exhibits strong extensibility, allowing seamless integration with various DQC-oriented compilation and optimization techniques. As these supporting technologies continue to advance, *HiQ-DF* is expected to generate more robust quantum circuits, further enhancing the problem-solving capabilities of QAOA in practical settings.

In future work, we plan to extend our research to distributed compilation techniques specifically for QAOA, such as advanced distributed qubit routing algorithms, and integrate them into *HiQ-DF* to build a complete end-to-end QAOA optimization toolchain for DQC. Moreover, we will further optimize the circuit design algorithms to address practical challenges arising from real quantum chip networks, such as gate fidelity and crosstalk, with the goal of improving the practical performance of QAOA.

Acknowledgements

Project supported by the National Natural Science Foundation of China (Grant No. 62472175), Shanghai Trusted Industry Internet Software Collaborative Innovation Center, and the "Digital Silk Road" Shanghai International Joint Laboratory of Trustworthy Intelligent Software (Grant No. 22510750100).

References

[1] Arute F, Arya K, Babbush R, *et al*. 2019 *Nature* **574** 505
[2] Quantum G A, Collaborators, Arute F, *et al*. 2020 *Science* **369** 1084
[3] Zhong H S, Wang H, Deng Y H, *et al*. 2020 *Science* **370** 1460
[4] Gao D, Fan D, Zha C, *et al*. 2025 *Phys. Rev. Lett.* **134** 090601
[5] Vikstål P, Grönkvist M, Svensson M, Andersson M, Johansson G and Ferrini G 2020 *Phys. Rev. Appl.* **14** 034009
[6] Sack S H and Egger D J 2024 *Phys. Rev. Res.* **6** 013223
[7] Khairy S, Shaydulin R, Cincio L, Alexeev Y and Balaprakash P 2020 *Proceedings of the AAAI Conference on Artificial Intelligence* **34** 2367
[8] Farhi E, Goldstone J and Gutmann S 2014 arXiv: 1411.4028 [quant-ph]
[9] Farhi E and Harrow A W 2016 arXiv: 1602.07674 [quant-ph]
[10] Guerreschi G G and Matsuura A Y 2019 *Sci. Rep.* **9** 6903
[11] Zhou L, Wang S T, Choi S, Pichler H and Lukin M D 2020 *Phys. Rev. X* **10** 021067
[12] Bharti K, Cervera-Lierta A, Kyaw T H, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann J S, Menke T, *et al.* 2022 *Rev. Mod. Phys.* **94** 015004
[13] Bechtold M, Barzen J, Leymann F, Mandl A, Obst J, Truger F and Weder B 2023 *Quantum Sci. Technol.* **8** 045022
[14] De Leon N P, Itoh K M, Kim D, Mehta K K, Northup T E, Paik H, Palmer B, Samarth N, Sangtawesin S and Steuerman D W 2021 *Science* **372** eabb2823
[15] Zhao P, Linghu K, Li Z, Xu P, Wang R, Xue G, Jin Y and Yu H 2022 *PRX Quantum* **3** 020301
[16] Zhong Y, Chang H S, Bienfait A, Dumur É, Chou M H, Conner C R, Grebel J, Povey R G, Yan H, Schuster D I and Cleland A N 2021 *Nature* **590** 571
[17] Niu J, Zhang L, Liu Y, *et al*. 2023 *Nat. Electron.* **6** 235
[18] Main D, Drmota P, Nadlinger D, Ainley E, Agrawal A, Nichol B, Srinivas R, Araneda G and Lucas D 2025 *Nature* **638** 383
[19] Harrigan M P, Sung K J, Neeley M, *et al*. 2021 *Nat. Phys.* **17** 332
[20] Alam M, Ash-Saki A and Ghosh S 2020 *Proceedings of the 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* pp. 215–228
[21] Jang E, Ha D, Choi S, Kim Y, Kwon J, Lee Y, Ahn S, Kim H and Ro W W 2024 *Proceedings of the 2024 International Conference on Parallel Architectures and Compilation Techniques (PACT)* pp. 309–324
[22] Zhu Y, Zhou Y, Cheng J, Jin Y, Li B, Niu S and Liang Z 2024 *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* pp. 1–7
[23] Einstein A, Podolsky B and Rosen N 1935 *Phys. Rev.* **47** 777
[24] Wu A, Zhang H, Li G, Shabani A, Xie Y and Ding Y 2022 *Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)* pp. 1027–1041
[25] Herrman R, Lotshaw P C, Ostrowski J, Humble T S and Siopsis G 2022 *Sci. Rep.* **12** 6781
[26] Wurtz J and Love P J 2022 *Quantum* **6** 635
[27] Chandarana P, Hegade N N, Paul K, Albarrán-Arriagada F, Solano E, Del Campo A and Chen X 2022 *Phys. Rev. Res.* **4** 013141

[28] Bonet-Monroig X, Wang H, Vermetten D, Senjean B, Moussa C, Bäck T, Dunjko V and O'Brien T E 2023 *Phys. Rev. A* **107** 032407

[29] Akshay V, Rabinovich D, Campos E and Biamonte J 2021 *Phys. Rev. A* **104** L010401

[30] Graham T, Song Y, Scott J, *et al*. 2022 *Nature* **604** 457

[31] Lacroix N, Hellings C, Andersen C K, Paolo A D, Remm A, Lazar S, Krinner S, Norris G J, Gabureac M, Heinsoo J, Blais A, Eichler C and Wallraff A 2020 *PRX Quantum* **1** 020304

[32] Proietti M, Cerocchi F and Dispenza M 2022 *Phys. Rev. A* **106** 022437

[33] Alam M, Ash-Saki A and Ghosh S 2020 *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC)* pp. 1–6

[34] Li J, Alam M and Ghosh S 2022 *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **42** 1852

[35] Kim S, Luo T, Lee E and Suh I S 2024 arXiv: 2407.20212 [quant-ph]

[36] Liaqat A, Darwish A, Roman A and DiAdamo S 2025 *IEEE International Conference on Quantum Software (QSW)* pp. 195–205

[37] Liu K, Zhou Y, Luo H, Xiong L, Zhu Y, Casey E, Cheng J, Chen SYC and Liang Z 2025 *IEEE International Symposium on Circuits and Systems (ISCAS)* pp. 1–5

[38] Ferrari D, Carretta S and Amoretti M 2023 *IEEE Trans. Quantum Eng.* **4** 1

[39] Mao Y, Liu Y and Yang Y 2023 *Proceeding of the 42nd IEEE Conference on Computer Communications (INFOCOM)* pp. 1–10

[40] Andres-Martinez P, Forrer T, Mills D, Wu J Y, Henaut L, Yamamoto K, Murao M and Duncan R 2024 *Quantum Sci. Technol.* **9** 045021

[41] Chen Z, Guan Z, Zhao S and Cheng X 2025 *Chin. Phys. B* **34** 050305

[42] Zhang S X, Hsieh C Y, Zhang S and Yao H 2022 *Quantum Sci. Technol.* **7** 045023

[43] Ferrari D, Cacciapuoti A S, Amoretti M and Caleffi M 2021 *IEEE Trans. Quantum Eng.* **2** 1

[44] Luo T Y, Zheng Y Z, Fu X and Deng Y X 2024 *Chin. Phys. B* **33** 120302

[45] Nielsen M A and Chuang I L 2010 *Quantum computation and quantum information* (Cambridge University Press)

[46] Bennett C H, Brassard G, Crépeau C, Jozsa R, Peres A and Wootters W K 1993 *Phys. Rev. Lett.* **70** 1895

[47] Barenco A, Bennett C H, Cleve R, DiVincenzo D P, Margolus N, Shor P, Sleator T, Smolin J A and Weinfurter H 1995 *Phys. Rev. A* **52** 3457

[48] Fowler A G, Devitt S J and Hollenberg L C 2004 arXiv: 0402196 [quant-ph]

[49] Li G, Ding Y and Xie Y 2019 *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* pp. 1001–1014

[50] Lao L and Browne D E 2022 *Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA)* pp. 351–365

[51] Lao L, Murali P, Martonosi M and Browne D 2021 *Proceedings of the 48th Annual International Symposium on Computer Architecture (ISCA)* pp. 846–859

[52] Wu A, Ding Y and Li A 2023 *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* pp. 479–493

[53] Brélaz D 1979 *Communications of the ACM* **22** 251

[54] Floyd R W 1962 *Communications of the ACM* **5** 345

[55] Mathur N, Landman J, Li Y Y, Strahm M, Kazdaghli S, Prakash A and Kerenidis I 2021 arXiv: 2109.01831 [quant-ph]

[56] Deng H, Zhang Y and Li Q 2020 *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC)* pp. 1–6