SWAP Optimization for Qubit Mapping Based on the Centric-Shortest Quantum Gate Set in NISQ Devices

Shujuan Liu[1], Hui Li[2]*, Yingsong Ji[3], Jiepeng Wang[4]
School of Computer and Information Engineering, Harbin University of Commerce, Harbin, China [1, 2, 3, 4]
Heilongjiang Provincial Key Laboratory of Electronic Commerce and Information Processing, Harbin, China [2]

***Abstract*—In the Quantum computing era of Noisy Intermediate-Scale Quantum (NISQ) devices, conventional qubit mapping strategies typically rely on specific heuristic rules to solve the mapping problem, overlooking the impact of other factors on the mapping, which leads to increased overhead from extra SWAP gates. To address this issue, we propose a SWAP optimization strategy based on the Centric-Shortest Quantum Gate Set (C-SQGS) and applies it to qubit mapping. In this approach, the centric qubit is determined by analyzing the maximum flexibility qubit set and the physical distances between the associated CNOT gates, leading to the identification of the Centric-Shortest Quantum Gate Set. To overcome the limitations of traditional cost functions that consider only single factors, a multi-factor cost function is introduced to evaluate the overall overhead of candidate SWAP operations and determine pending SWAP gate Set. Based on qubit flexibility analysis, executable SWAP gate is identified and inserted into the circuit. Experimental results demonstrate that the C-SQGS strategy effectively reduces both SWAP gate and two-qubit gate overhead. Specifically, it achieves an average SWAP gate reduction of 36.9% and 47.7%, and a two-qubit gate reduction of 13.8% and 13.5% on the t|ket⟩ and Qiskit compilers, respectively. These results highlight the potential of the C-SQGS strategy in enhancing the efficiency of qubit mapping for NISQ devices.**

***Keywords*—Quantum computing; qubit mapping; Centric-Shortest Quantum Gate Set (C-SQGS); executable SWAP gate; multi-factor cost function**

I. Introduction

Quantum computing is an innovative approach to computation that utilizes the principles of quantum mechanics to process information. In contrast to classical computers that use bits, quantum computing operates with qubits. Qubits can simultaneously exist and the superposition of states, this superposition property makes the quantum computer can simultaneously deal with a variety of possible states, thus accelerating the process of solving complex problems. The powerful parallel computing acceleration capability of quantum computing gives it great potential in areas such as integer factorization [1], optimization problems [2], and quantum simulation [3]. Due to the inherent limitations of traditional computers, it can be time-consuming and inefficient in dealing with these more complex problems, while quantum computers provide new solutions to these problems.

As quantum computing technology advances rapidly, it has now transitioned into the NISQ era. Executing a quantum logic circuit involves aligning the algorithm's logical qubits with the physical qubits of the quantum computer, a process called qubit mapping [4]. Due to the limitation of the technology in the NISQ era, two-qubit gate operations can only be performed on physically coupled qubit pairs. Thus, during the mapping process, quantum circuits must be adjusted to ensure compatibility with the hardware's coupling constraints. For two-qubit gates that do not comply with the hardware coupling constraints, extra SWAP gates must be added to reposition the qubits to adjacent locations. The additional SWAP operation, however, not only increases the depth of the quantum circuit, but also introduces more noise. Thus, minimization of the count of extra SWAP gates has become the focus of many researchers [5]-[8].

The qubit mapping problem has been demonstrated to be NP-complete [9]-[10]. In recent years, domestic and foreign scholars have carried out many studies on heuristic qubit mapping algorithms. Siraichi [11] proposed to preferentially satisfy physically neighboring CNOT gates during the initial mapping, and to resolve only one two-qubit gate operation each time the qubits are shifted. Although this method is relatively fast, it is generally ineffective in coping with complex quantum circuits due to the relative simplicity of the strategy. Zulehner [12] used the A* algorithm to determine the optimal mapping of each logical to physical qubit to optimize the implementation of quantum circuits on hardware architectures, but due to the exhaustive search, the running time is long. In 2020, Adrien [4] proposed the Hardware-Aware (HA) algorithm to decrease the quantity of SWAP gates, though it allows for the selection between SWAP and Bridge gates, it does not adequately account for the effect of varying numbers of look-ahead gates on the circuit's cost. Steinberg [13] proposed the HQAA algorithm in 2022 for assigning initial qubit positions for quantum algorithms on NISQ devices, but it fails to achieve adequate efficiency in large-scale quantum circuits. The above literature, while proposing valuable treatments on the mapping problem, still requires further improvements when dealing with complex quantum circuits.

In the NISQ era, the existing quantum circuit mapping algorithms not only include the initial qubit mapping but also involve the intermediate change strategy of qubit mapping. After the initial mapping is completed, a reasonable intermediate change strategy is formulated based on the information of the quantum circuit, which helps to improve the performance of the overall quantum circuit. Due to hardware topology constraints, the initial mapping does not meet the coupling requirements for all qubits, and for the non-Nearest

*Corresponding author
This work was supported by the Natural Science Foundation of Heilongjiang Province of China [Grant Number: LH2024F042].

Neighbor (non-NN) gates that do not satisfy the constraints, it is necessary to implement an additional SWAP operation to exchange qubits to make them update to Nearest Neighbor (NN) gates, and this process will expand the size and depth of the circuit. Two-qubit gates have significantly higher error rates than single-qubit gates [14]. Hence, minimizing the compilation overhead is crucial for high fidelity circuit implementations.

This paper focuses on the qubit mapping problem subject to physical topology constraints. A SWAP optimization strategy based on Centric-Shortest Quantum Gate Set (C-SQGS) is proposed, aiming to optimize quantum circuits for the 2-local qubit Hamiltonian simulation problem. This strategy optimizes the initial mapping of quantum circuits by identifying the Centric-Shortest Quantum Gate Set, and employs a multi-factor cost function together with qubit flexibility analysis to search for optimal SWAP gates. By comprehensively considering multiple factors, the proposed method effectively minimizes the number of inserted SWAP gates.

Unlike existing qubit mapping and routing heuristics that rely on single-factor optimization or local greedy decisions [23], the C-SQGS strategy introduces a centric decision paradigm that organizes SWAP insertion around a dynamically identified centric qubit and its associated shortest gate set. Rather than independently applying concepts such as qubit flexibility or gate frequency, C-SQGS integrates these factors into a unified gate-set–driven mapping framework, enabling coordinated optimization across both the initial mapping and routing stages.

From a practical perspective, qubit mapping on NISQ devices is constrained not only by physical connectivity but also by limited coherence time and restricted scheduling flexibility. Existing heuristics often rely on locally optimal or single-metric decisions, which may introduce SWAP operations that are executable in isolation but suboptimal or even harmful when considering subsequent circuit evolution. The proposed C-SQGS framework, together with executable SWAP identification and multi-factor cost evaluation, is designed to address these practical limitations by enabling globally informed and execution-aware SWAP insertion.

The C-SQGS strategy is able to adapt to a variety of qubit topologies and gate sets and can be used in conjunction with a wide range of existing quantum circuit mapping algorithms. Experimental findings indicate that the C-SQGS strategy somewhat reduces the count of SWAP and two-qubit gates. Moreover, we conduct experiments on IBM devices to verify the superiority of our strategy and compare it with other algorithms.

This paper makes the following key contributions:

- We focus on the qubit mapping challenge and point out the main influencing factors considered in this paper.
- A SWAP optimization strategy based on a Centric-Shortest Quantum Gate Set is proposed.
- Designing a multi-factor cost function that considers multiple factors simultaneously.
- Introduction of a selection criterion and the definition for executable SWAP gates.

The remainder of this paper is structured as follows. Section II outlines essential background on quantum computing. Section III highlights the challenges in qubit mapping and introduces the proposed approach. Section IV details the experimental evaluation, and Section V summarizes the conclusions.

II. BACKGROUND

A. Prior Knowledge

In quantum computing, a qubit is the core unit that can simultaneously occupy the $|0\rangle$ and $|1\rangle$ states through the principle of superposition. The quantum state can be expressed as Eq. (1), where $|\psi\rangle$ is the state vector of the qubit, α and β are complex numbers representing the probability of the qubit being in the states $|0\rangle$ and $|1\rangle$, respectively, and satisfying the condition $|\alpha|^2+|\beta|^2=1$, whereas $|0\rangle$ and $|1\rangle$ are the ground states of the qubit. This special property of qubits enables quantum computers to achieve exponential speedups.

$$|\psi\rangle=\alpha|0\rangle+\beta|1\rangle \tag{1}$$

Quantum gates are the fundamental units of operation in quantum computing, used to manipulate the state of qubits. Fig. 1 shows the symbolic representation of basic quantum gates. Among them, the Hadamard gate is a single-qubit operation that transforms a classical state into an equal superposition of basis states. Whereas CNOT and SWAP gates are gates acting on two qubits, CNOT gates can realize a kind of qubit control flip-flop operation, which passes the state of one qubit to the other qubit and thus establishes an entanglement between them. SWAP gates consist of 3 CNOT gates, as in Fig. 1(c), which can realize the exchange of states between qubits without changing their relative phase.

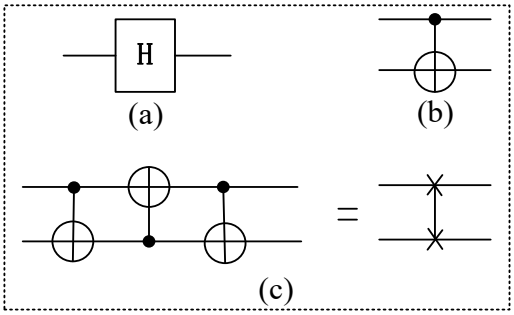

Fig. 1. Basic quantum gates.

Quantum circuit is a fundamental computational model in quantum computing for describing the process of manipulating qubits through quantum gates. Fig. 2 shows a simple quantum circuit describing the whole process of measuring the final state of a qubit after the initial state of the qubit passes through the quantum circuit, which consists of a Hadamard, a CNOT and a SWAP gate, and the essence of the quantum circuit is a combination of the unitary transformation and measurement. Physically, we cannot directly realize the overly complex unitary transformations, so expect some easy to realize the

unitary transformations to produce more complex unitary transformations. For complex quantum operations, such as Hamiltonian simulation, can be realized through the rational design and adjustment of quantum gates within the quantum circuit.

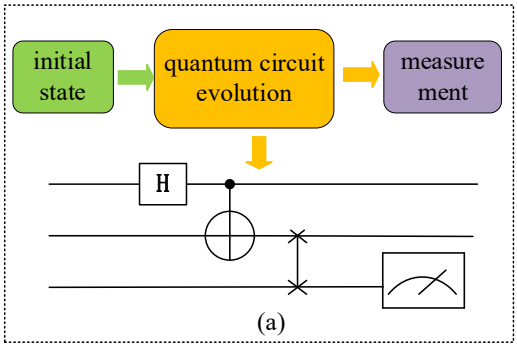

Fig. 2. Quantum circuit.

B. Hamiltonian

The Hamiltonian operator characterizes the total energy of a system. In quantum mechanics, the Hamiltonian acts on the quantum state of a system, describing its energy state and time evolution. In quantum computing, the Hamiltonian is used to represent the energy states of qubits and the interactions between them. Quantum gate operations can be implemented through time evolution operators [see Eq. (2)], where $\hat{H}$ is the Hamiltonian corresponding to the quantum gate.

$$U(t) = e^{-\frac{i\hat{H}t}{\hbar}} \tag{2}$$

Hamiltonian simulation is a vital application in quantum computing, involving the modeling of the Hamiltonian of a quantum system. By simulating Hamiltonian dynamics, one can predict and analyze the time evolution of quantum systems, with applications in quantum physics, chemistry, materials science, and beyond. Various methods have been proposed for effective Hamiltonian simulation, including Taylor series expansions [15], the product formula [16]-[17], and quantum signal processing [18]-[19]. Among these, the product formula is particularly notable for its simplicity and practical efficacy [20]. The Hamiltonian of a system, denoted as H, can be decomposed into a sum of polynomial ergodic terms H_j, as illustrated in Eq. (3). In quantum computing, the product formula is a vital tool for describing time-evolution operators, especially for time-dependent Hamiltonians. This method represents the time-evolution operator as a product of a series of simpler operators, thereby simplifying the implementation of complex quantum operations on a quantum computer. The product formula is detailed in Eq. (4).

$$H = \sum_{j=1}^{L} h_j H_j \tag{3}$$

$$V(t) = \prod_{j=1}^{L} \exp\left(ith_j H_j\right) \tag{4}$$

where, $\exp\left(ith_j H_j\right)$ represents the time evolution caused by the Hermitian term H_j, i is the imaginary unit, t is the time, h_j is the coefficient of each Hermitian term H_j, L is the number of Hermitian terms into which the system Hamiltonian H is decomposed, and $V(t)$ represents the total evolution operator of the system at moment t. Due to the flexibility of the arrangement of Hamiltonian operators, it is possible to optimize the Hamiltonian analog circuit for each Trotter step by rearranging L individual operators in the product formula, thus optimizing the overall circuit structure.

III. Our Design

A. Limitations of Qubit Mapping

Qubit mapping is an important process in quantum computing aimed at mapping abstract quantum algorithms or quantum circuits onto concrete quantum hardware for practical execution. This process involves several steps, including mapping logical to physical qubits, optimizing the sequence of quantum gate operations, minimizing SWAP operations, etc. Due to the limitations of NISQ era technology, execution is only possible on qubits that are allowed to be connected on quantum hardware coupling architectures. In executing quantum logic circuits, logical qubits are required to match physical quits. The limitations of quantum mapping are further explained with examples as shown in Fig. 3. Fig. 3(b) shows a small quantum circuit which consists of 7 CNOT gates and 1 Hadamard gate and is executed on the 6-qubit device shown in Fig. 3(a). The device allows the use of two-qubit gates on the qubit pairs $\{Q_1, Q_2\}$,$\{Q_2, Q_3\}$,$\{Q_3, Q_4\}$,$\{Q_4, Q_5\}$,$\{Q_0, Q_5\}$,$\{Q_0, Q_1\}$,$\{Q_2, Q_5\}$ and $\{Q_3, Q_4\}$, and does not allow the use of two-qubit gates on the qubit pairs $\{Q_1, Q_5\}$,$\{Q_0, Q_2\}$,$\{Q_2, Q_4\}$ and $\{Q_3, Q_5\}$. For simplicity, assume that the initial mapping is $\{q_0 \rightarrow Q_0, q_1 \rightarrow Q_1, q_2 \rightarrow Q_2, q_3 \rightarrow Q_3, q_4 \rightarrow Q_4, q_5 \rightarrow Q_5\}$. Due to the limited connectivity between qubits, only directly connected qubits can perform quantum gate operations. From Fig. 3(a), among the seven CNOT gates in Fig. 3(b), g_1, g_2, g_3 and g_5 can be directly executed. Since the qubits associated with the three CNOT gates, g_4, g_7 and g_8, do not have a direct mapping to the actual hardware, g_4, g_7 and g_8 cannot be directly executed, and a change in the qubit mapping is required before executing these two gates.

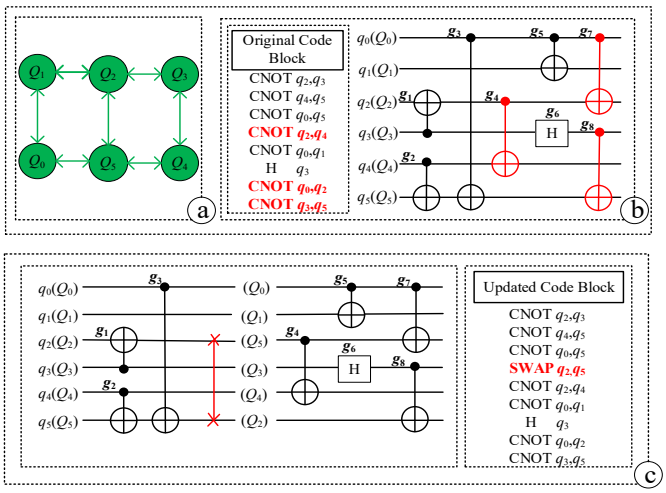

Fig. 3. (a) Hardware topology, (b) Original quantum circuit, (c) Updated quantum circuit.

The objective of qubit mapping is to identify the optimal initial qubit placement to minimize the count of SWAP operations required when converting all non-NN gates to NN gates. The qubit mapping challenge is modeled as a Quadratic Assignment Problem (QAP) [21]. For solving this problem, the qubit positions are changed by inserting SWAP gates by exchanging the state between two qubits [22]. In Fig. 3(c), CNOT gates g_1, g_2, g_3 and g_5 can be executed with the initial mapping, with SWAP$\{q_2,q_5\}$ inserted after g_3, the mapping is updated to $\{q_0 \rightarrow Q_0, q_1 \rightarrow Q_1, q_2 \rightarrow Q_5, q_3 \rightarrow Q_3, q_4 \rightarrow Q_4, q_5 \rightarrow Q_2\}$, which exchanges the two qubits q_2, q_5 and makes the subsequent gates g_4 and g_7 into executable NN gates.

The insertion of extra SWAP gates can generate hardware-compatible quantum circuits, but the increase in SWAP gates may introduce additional noise that reduces the fidelity of the quantum circuits, which in turn affects the overall performance of the circuit. In addition, the increase in line depth can extend the execution time and negatively affect the efficiency of quantum computation. Comparing the original and updated circuits in Fig. 3(b) and Fig. 3(c), the count of CNOT gates increases from 7 to 10, and the depth grows from 5 to 8. The addition of these gates introduces extra overhead, impacting both the fidelity and execution time of the quantum circuits. This paper is devoted to exploring the optimal solution of the qubit mapping challenge, with a view to minimizing the overhead of extra SWAP gates and effectively improving mapping quality.

The qubit mapping stage is a crucial step in quantum computing, where the mapping of logical qubits to physical qubits directly impacts the execution efficiency and computational accuracy of quantum circuits. So far, there are still many challenges in the qubit mapping stage, and the following limitations are mainly considered in this paper:

- Limitations of hardware topology: The way qubits are connected to each other and the operational limitations are determined by the hardware design. These constraints include the layout of qubits, the way they interact with each other, and the execution efficiency of quantum gates. Reasonable mapping of quantum circuits to the actual hardware topology can maximize the execution efficiency and accuracy of the algorithm.
- Limited connectivity between qubits: The connections between physical qubits on a quantum processor are usually limited. Not all qubits are directly connected, and only neighboring qubits can directly perform two-qubit gate operations. This limitation requires moving qubits around by adding extra SWAP operations that allow non-directly connected qubits to interact.
- SWAP gate operation cost: One SWAP gate is equivalent to 3 CNOT gates, and the operation cost of SWAP gates is generally higher. Therefore, when addressing the qubit mapping challenge, we must consider the operation cost of SWAP gates and select the SWAP insertion with lower operation cost.
- Balancing considerations for optimizing performance: The goal of qubit mapping is the minimization of the count of SWAP gate operations when all two-qubit gates are converted from non-NN gates to NN gates. Therefore, when dealing with the qubit mapping challenge, different performance metrics need to be considered, such as the starting point of the initial mapping, the design of the cost function, the cost of SWAP gate operations, and the selection of SWAP gates.

B. SWAP Optimization Strategy Based on Centric-Shortest Quantum Gate Set

In general, it is difficult to find a perfect initial mapping such that it satisfies the dependencies of all two-qubit gates. For non-NN gates, additional SWAP operations are required to convert them to NN gates, a process that leads to overhead in gate counting and circuit depth. Therefore, efficient qubit routing techniques are essential to minimize compilation overhead for reliable computation. Lao [23] introduced a quantum compiler, 2QAN, in 2022, which mainly optimizes the 2-local quantum simulation problem, such as the Quantum Approximation Optimization Algorithm (QAOA). It leverages flexibility in arranging Hamiltonian quantum operations and reduces two-qubit gate overhead by integrating SWAP gates with circuit gates into a single unitary transformation. Although 2QAN presents a meaningful solution to qubit mapping challenge, it neglects the effect of the number of qubits involved in the CNOT operation on the mapping process. It has been found that prioritizing the mapping of qubits involved in a high number of CNOT gates to locations close to each other reduces the subsequent additional SWAP gates, thus reducing the circuit depth and execution time [24]. Based on this, this paper proposes a SWAP optimization strategy based on the Centric-Shortest Quantum Gate Set(C-SQGS).

Definition 1 Qubit Flexibility: In a quantum circuit C, let q_i be any qubit whose number $F(q_i)$ of times it participates in a non-NN gate operation is q_i's Qubit Flexibility.

Definition 2 Maximum Flexibility Qubit Set: In a quantum circuit C, assuming $G_n = \{g_1, g_2, \ldots\ldots, g_n\}$ be the set of non-NN gates and $Q_m = \{q_1, q_2, \ldots\ldots, q_m\}$ be the set of qubits involved in the quantum gates in G_n. The Qubit Flexibility Set is $F_Q = \{F(q_i), i = 1,2,\ldots\ldots,m\}$. If $F_{max}(q_j) = \max F_Q$, then $q = \{q_j, j = 1,2,\ldots\ldots,m\}$ is said to be the Maximum Flexibility Qubit Set in Q_m.

Definition 3 Centric Qubit: In a quantum circuit C, assuming $G = \{G_1, G_2, \ldots\ldots, G_m\}$ be the set of quantum gate sets associated with q_j. The set of physical spacings of the corresponding qubits in the set of quantum gates in G is $D = \{D(G_1), D(G_2), \ldots\ldots, D(G_m)\}$. If $D(G_k) = \min D$, then q_k is said to be the Centric Qubit in Q_m, where, $j = 1,2,\ldots\ldots,m$; $k = 1,2,\ldots\ldots,m$.

Definition 4 Centric-Shortest Quantum Gate Set: In a quantum circuit C, assuming the set of quantum gates associated with the Centric Qubit q_k be $G_k = \{g_1, g_2, \ldots\ldots, g_r\}$, and the set of physical spacings of the corresponding qubits for each quantum gate in G_k be $D' = \{D(g_1), D(g_2), \ldots\ldots, D(g_r)\}$. If

$D(g_z) = \min D'$, and then $G_k' = \{g_z, z = 1,2,\ldots\ldots,r\}$ is said to be the Centric-Shortest Quantum Gate Set for the quantum circuit C where, $0 < z \le r \le n$.

The SWAP optimization strategy based on Centric-Shortest Quantum Gate Set (C-SQGS) is a method for optimizing quantum circuit mapping. It aims to identify the Centric-Shortest Quantum Gate Set in quantum circuits, minimize the use of SWAP gates and reduce communication overhead between qubits, and enhance the efficiency of the mapping process. For non-NN gates, the specific operation steps are as follows:

Step 1: Based on the concept of Qubit Flexibility, determine Maximum Flexibility Qubit Set, denoted as q.

Step 2: If q > 1, then the Centric qubit q_k is further determined according to Definition 3.

Step 3: G_k is the set of quantum gates associated with q_k, calculate the physical distance between the corresponding qubits of each of these CNOT gates, prioritize these CNOT gates based on the distances, and identify the CNOT gates with the shortest spacing, so as to determine the Centric-Shortest Quantum Gate Set G_k', and make sure that it is prioritized.

Step 4: The cost function (Definition 5) is used to evaluate all candidate SWAP operations and select the pending SWAP gate set (Definition 6).

Step 5: According to Definition 7, evaluate the pending SWAP gate set and determine the executable SWAP gates, which are inserted into the quantum circuit.

Step 6: Update the mapping to eliminate the newly generated NN gates from the set of non-NN gates.

Step 7: Repeat Step1- Step6 until all gates are mapped.

The C-SQGS strategy deals with the quantum circuit mapping challenge by optimizing the selection and arrangement of quantum gates, thus reducing the complexity of the SWAP gate search space, reducing the SWAP gate overhead, and improving the overall performance of quantum computing. Algorithm 1 is the detail of the pseudocode.

Algorithm 1 Optimization algorithm

Input: Un-routed quantum circuit, initial mapping M

Output: Routed quantum circuit, NN gates

```
begin
  Initialize the set of NN gates for each mapping, G_NN = {G_M}, G_M ← all NN gates for mapping M
  Initialize G_non-NN ← all un-routed Non-NN gates
  Initialize Q={q_i} ← all qubits involved in Non-NN gates
  while G_non-NN≠∅ do
    q ← set of qubits with Maximum Flexibility Qubit Set in Q
    if q >1
      Select the Centric Qubit q_k∈Q in q
      Select the gate G'_k∈G_non-NN that has Centric-Shortest Quantum Gate Set in G_non-NN
      S_g ← SWAP gates with the identical cost
      S_b ← Select the executable SWAP gate from S_g
      G'_NN ← Find NN gates in G_non-NN for map M'
      Remove all gates in G'_NN from G_non-NN, add G'_NN to G_NN
    end if
  end while
end
```

C. Multi-Factor Cost Function Design

The design of the cost function is particularly important when dealing with qubit mapping challenges, and a reasonable cost function can ensure the executability of quantum circuits. However, the traditional cost function usually takes the distance between qubits as the core criterion of measurement, which lacks the comprehensive consideration of multiple factors and makes it difficult to accurately assess the actual overhead of SWAP gates. To solve this problem, a multi-factor cost function is designed in this paper.

Definition 5 Multi-Factor Cost Function: The minimization of the sum of the weighted products of the distance between qubits, the number of interactions, and the interaction time is shown in Eq. (5).

$$H_{cost} = \min_{\phi \in S_n} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} F_{q_i q_j} D_{[\phi(q_i)\phi(q_j)]} R_{q_i q_j} \right\} \tag{5}$$

where, S_n represents the set of all possible permutations, $F_{q_i q_j}$ is the interaction time between circuit qubits q_i and q_j, $R_{q_i q_j}$ is the number of interactions between qubits q_i and q_j in the circuit, and $D_{[\phi(q_i)\phi(q_j)]}$ denotes the distance between physical qubits on the hardware $\phi(q_i)$ and $\phi(q_j)$, calculated using the Floyd-Warshall algorithm.

The interaction time between qubits affects the execution speed of the circuit, the shorter the interaction time between qubits, the faster the circuit executes, and longer interaction times lengthen the overall execution time of the quantum circuit [25]. The number of interactions is also another important factor affecting the mapping of quantum circuits [26], each quantum gate operation introduces errors, a higher number of interactions leads to more operations, increasing the risk of error accumulation and decreasing the accuracy and reliability of the computation. Reducing the number of interactions helps to reduce the accumulation of errors and improve the accuracy of the calculation. In quantum circuits, CNOT gates need to be executed between directly connected qubits. If two qubits are physically distant from each other, multiple intermediate SWAP operations need to be introduced to make them physically adjacent. These additional operations increase the depth and execution time of the circuit and affect the overall performance of the quantum algorithm. In this paper, we combine these three factors to design a Multi-Factor

Cost Function that can handle the qubit mapping challenge more comprehensively. For non-NN gates, the cost function can assign a value to each candidate SWAP gate to measure the advantages and disadvantages of each operation and filter out the optimal SWAP operation. The mapping layout of qubits needs to be re-evaluated and adjusted after each SWAP operation.

D. Executable SWAP Gate

In the field of quantum computing, CNOT gates are crucial, and together with any single-qubit gate, they form a universal set of quantum gates. This means that any complex quantum circuit can be realized with the appropriate combination of these fundamental gates. When dealing with the qubit mapping challenge, some of the CNOT gates do not satisfy the conditions for execution in the initial mapping because the qubit arrangement and interactions in a real quantum computer are limited by the hardware topology. To make all CNOT gates executable, extra SWAP gates need to be inserted so that the control and target bits of the CNOT gates satisfy the conditions of the hardware coupling constraints. However, extra SWAP gates impose additional overhead on the circuit. The identification of executable SWAP operations from all candidate SWAP gates is key to the considerations in this paper.

Definition 6 Pending SWAP Gate Set: The set of SWAP operations that minimize the cost function by evaluating all candidate SWAP gates using the cost function is referred to as the Pending SWAP Gate Set.

Definition 7 Executable SWAP Gate: There are n qubits in quantum circuit C. Assuming $F_{\max}(q_i), i=(1,2,\ldots\ldots,n)$ be the Maximum Flexibility Qubit Set under the initial mapping M. $G_{SWAP}=\{g_1,g_2,\ldots\ldots,g_z\}$ is a list of Pending SWAP Gate set, and $F^{'}=\{F^{'}(q_r), r=1,2,\ldots\ldots,z\}$ denotes the Maximum Flexibility Qubit Set corresponding to a SWAP operation under the new mapping M' after the execution of G_{SWAP}. If $F^{'}_{\max}(q_j)=\max F^{'}$, then the SWAP gate g_j corresponding to q_j is said to be an Executable SWAP Gate, where , $1\le j\le z$.

To minimize the count of SWAP gates, firstly, Pending SWAP Gate Set is filtered by using the cost function (Definition 5); secondly, considering that highly flexible qubits usually have strong dependence on other qubits, based on Qubit Flexibility, the SWAP operation mode corresponding to maximum flexibility qubit is selected, so as to minimize the exchange distance between physical qubits and determine the Executable SWAP Gate. This selection strategy achieves the reduction of extra overhead and improves the overall performance of the quantum circuit. The specific procedure for determining an Executable SWAP Gate is as follows:

Step 1: Calculate the Maximum Flexibility Qubit Set $F_{\max}(q_i)$ under the initial mapping M.

Step 2: The simulation executes each potential SWAP gate to obtain the Maximum Flexibility Qubit Set $F^{'}$ corresponding to SWAP operations under the new mapping M'.

Step 3: Compare the values in $F^{'}$ to obtain the $F^{'}_{\max}(q_j)$ under the new mapping M'.

Step 4: Under the new mapping M', the SWAP gate corresponding to the largest value in the set $F^{'}$ is considered as an executable SWAP gate and applied to the circuit. By in this way, the count of extra SWAP operations can be reduced and the whole quantum circuit can be optimized.

The details of the pseudocode are presented in Algorithm 2, with a concrete example provided in Fig. 4.

Algorithm 2 Select the Executable_SWAP

Input: top_qubit_number: After inserting a SWAP gate, Maximum Flexibility Qubit Set under the new mapping M'.
moves: A Pending SWAP Gate Set
Output: Executable_SWAP

```
Begin
  max_top_qubit_number_add ← 0
  Executable_SWAP ← None
  for move in moves do
    top_qubit_ number_ add←top_qubit_ number [move]
      if top_qubit_numbe _add > max_top_qubit_ number_ add
        max_top_qubit_number_add ← top_qubit_ number_ add
        Executable_SWAP ← move
      end if
  end for
end
```

The IBM Q20 Tokyo localized architecture is shown in Fig. 4. Assume that the code block on Fig. 4(a) is executed on this architecture, where SWAP $\{q_0, q_4\}$ and SWAP $\{q_0, q_1\}$ are pending SWAP gates. According to as in Fig. 4(a), $F_{\max}(q_0)=2$ under the initial mapping M. Insert the candidate SWAP $\{q_0, q_4\}$ and SWAP $\{q_0, q_1\}$ into the quantum circuit to see the corresponding $F^{'}_{\max}(q_j)$ values under the updated mapping M', respectively. As in Fig. 4(b), insert SWAP $\{q_0, q_4\}$, under the current mapping M', $F^{'}_{\max}(q_0)=1$. As in Fig. 4(c), insert SWAP $\{q_0, q_1\}$, under the updated mapping M', $F^{'}_{\max}(q_9)=2$. The strategy in this paper is to find the SWAP operation that maximizes the $F^{'}_{\max}(q_j)$. By comparison, SWAP $\{q_0, q_1\}$ gives more benefits to the circuit than SWAP $\{q_0, q_4\}$, therefore, SWAP $\{q_0, q_1\}$ is chosen to insert and update the mapping. Fig. 4(d) shows the final mapping result graph obtained based on the SWAP strategy proposed in this paper, where only 2 extra SWAP operations need to be added to Fig. 4(c) to update all quantum gates to NN gates, SWAP $\{q_5, q_9\}$ and SWAP $\{q_2, q_3\}$. However, the method of Fig. 4(b) requires at least 3 SWAP gates to be inserted to update all quantum gates to NN gates. C-SQGS strategy increases qubit flexibility, and the increase in flexibility reduces the non-NN gate operations, allowing qubits to be closer to each other, thus optimizing the overall structure of the quantum circuit.

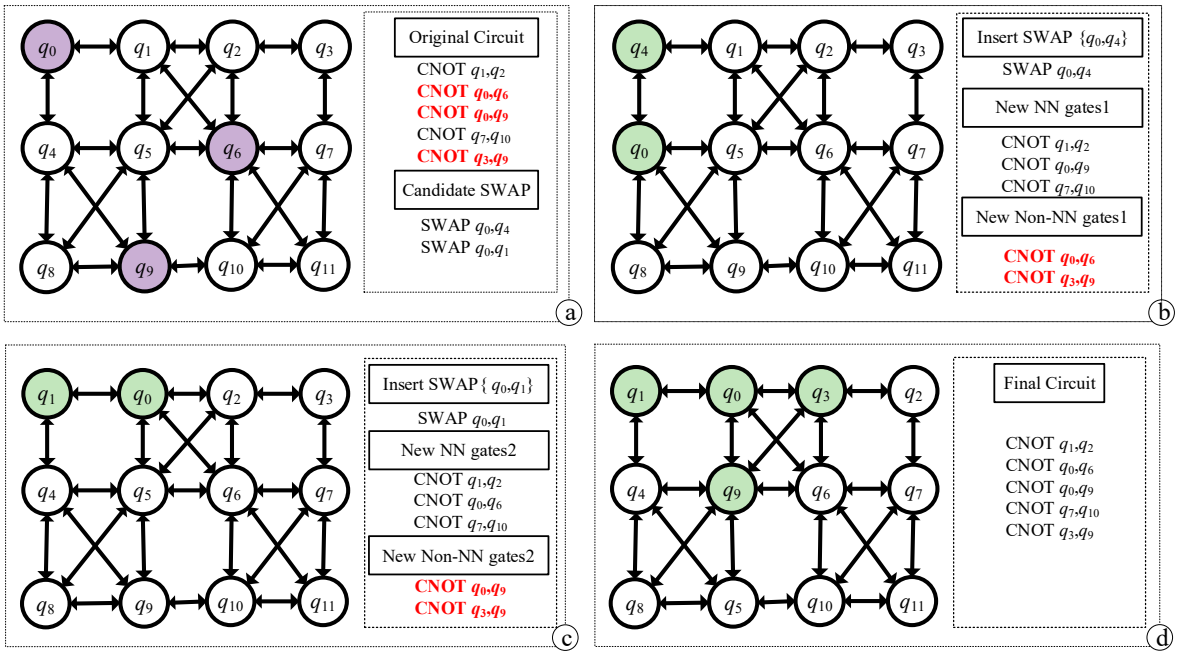

Fig. 4. (a) Original circuit, (b) Updated circuit after inserting SWAP $\{q_0, q_4\}$, (c) Updated circuit after inserting SWAP $\{q_0, q_1\}$, (d) Final circuit.

IV. Results Assessment

In this section, the main focus is on evaluating the operational results. Compared with one-dimensional chains and three-dimensional structures, two-dimensional structures have higher parallelism, qubits are arranged in the plane, and coupling and control are more simplified, which helps to reduce the noise level and error accumulation. Therefore, in this paper, we chose to compile on the IBM Quantum ibmq_montreal device with a two-dimensional structure, as shown in Fig. 5. The following key metrics are used to measure the effectiveness of the C-SQGS strategy on the compiler, one is the count of extra SWAP gates, and the other is the count of two-qubit gates, both of which are evaluated on the basis of the criterion that fewer is better [8]. The benchmark test program is IBM's Qiskit quantum program, which uses the Qiskit compiler to decompose and optimize the CNOT (CX) gate set. The benchmarking methodology of Tannu [27] was employed. The entire experiment was conducted using Python 3.11, with all compilations running on a laptop equipped with an Intel Core i5 processor (2.50 GHz, 8 GB RAM). Evaluations for 4 to 22 qubits were carried out, running the mapping process five times and selecting the best outcome.

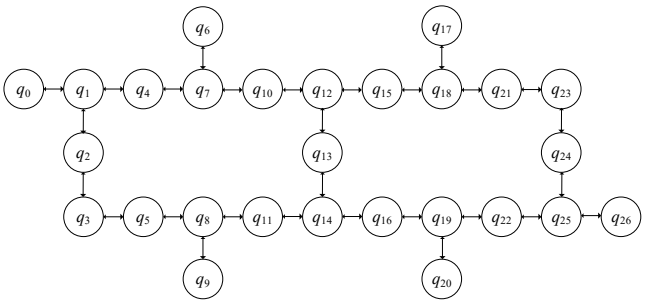

Fig. 5. Ibmq_montreal device.

In this paper, we compare the C-SQGS strategy with the compilation overhead of t|ket⟩ and Qiskit. The t|ket⟩ [28] and the Qiskit compiler [29] are equipped with the recommended "FullPass". The specific overheads of the C-SQGS with respect to the t|ket⟩ and Qiskit compilers are shown in Fig. 6. There are some optimizations in both the extra SWAP gate and CNOT gate counting overheads. The C-SQGS strategy is optimized relative to 2QAN, although the optimization is more significant for large qubits than for small qubits.

In the t|ket⟩ compiler, for circuits with 12-22 qubits, the C-SQGS strategy reduces the quantity of extra SWAP gates and CNOT gates by 6.1% and 5.5%, respectively, compared to 2QAN, as illustrated in Fig. 6(a), (b). For circuits with 4-10 qubits, the quantity of SWAP gates is decreased by an average of 26.4% and the quantity of CNOT gates by an average of 13.2% across all evaluated benchmarks after applying the C-SQGS strategy of this paper; the quantity of SWAP gates and CNOT gates are reduced by an average of 43.8% and 14.3%, respectively, after optimization of quantum circuits of 12-22 qubits, as illustrated in Fig. 7(a).

Within the Qiskit compiler framework, for circuits with 12-22 qubits, the C-SQGS strategy reduces the quantity of extra SWAP and CNOT gates by 5.7% and 7.6%, respectively, compared to 2QAN, as illustrated in Fig. 6(c), (d). For circuits with 4-10 qubits, after utilizing the C-SQGS strategy proposed in this paper, the quantity of SWAP gates is reduced by an average of 27.8% and the quantity of CNOT gates by an average of 14.9% in all the evaluated benchmarks, and the optimization of the quantum circuits of 12-22 qubits results in an average reduction of 61% in the quantity of SWAP gates, an average reduction in the quantity of CNOTs of 12.5%, and the overhead of double quantum gates is reduced by a factor of 2.5 compared to using only 2QAN, as shown in Fig. 7(b).

To further validate the advancement of the C-SQGS strategy, it is compared with the BSOS [8] and HQAA algorithms [13], as seen in Fig. 8. Within the t|ket⟩ compiler framework, for circuits with 4-10 qubits, the optimization effect of the C-SQGS is relatively tiny compared with the BSOS and HQAA algorithms, which is due to the fact that the circuit is simpler and the number of SWAP gates originally required is less in smaller quantum circuits, and the optimization effect is not easy to be seen. For circuits with 12-22 qubits, the advantage effect of the C-SQGS becomes more apparent. Specifically, compared with HQAA, the C-SQGS reduces the quantity of SWAP gates and CNOT gates by 9.3% and 12%, respectively. In the Qiskit compiler, for circuits with 4-10 qubit, C-SQGS reduces the quantity of CNOT gates by 4.1% and 4.3% compared to BSOS and HQAA, respectively. For circuits with 10-22 qubits, the quantity of SWAP gates is reduced by 3.1% and 12.7%, and the quantity of CNOT gates is reduced by 2% and 9.6%, respectively.

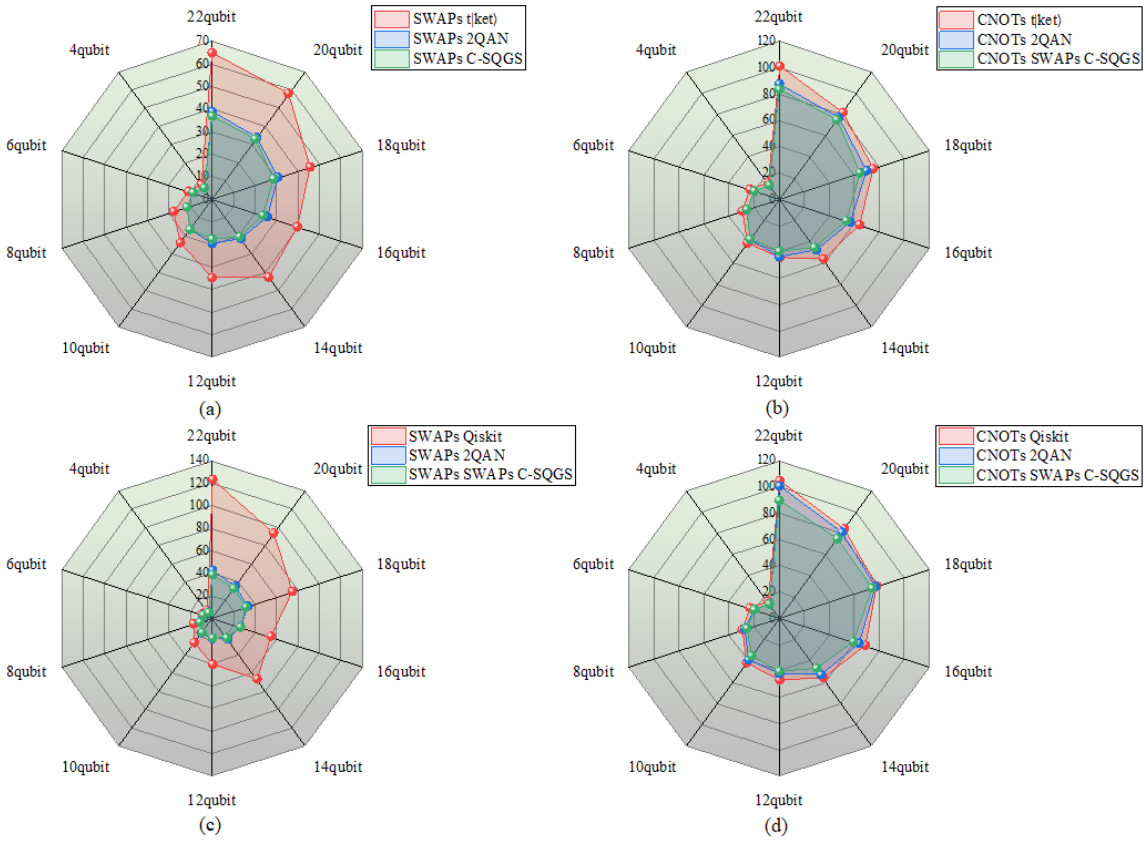

Fig. 6. (a) SWAP gate Overhead comparison between C-SQGS strategy, t|ket⟩ and 2QAN, (b) CNOT gate Overhead comparison between C-SQGS strategy, t|ket⟩ and 2QAN, (c) SWAP gate Overhead comparison between C-SQGS strategy, Qiskit and 2QAN, (d) CNOT gate Overhead comparison between C-SQGS strategy, Qiskit and 2QAN.

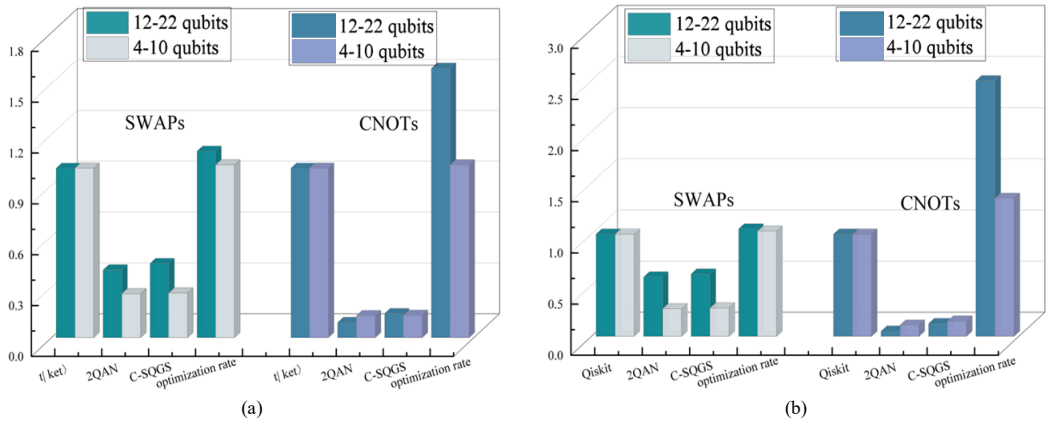

Fig. 7. The C-SQGS strategy was assessed for circuits with 4-10 qubits and 12-22 qubits. (a) Optimization ratio of SWAP and CNOT gates for C-SQGS compared to t|ket⟩ and 2QAN. (b) Optimization ratio SWAP and CNOT gate for C-SQGS compared to Qiskit and 2QAN.

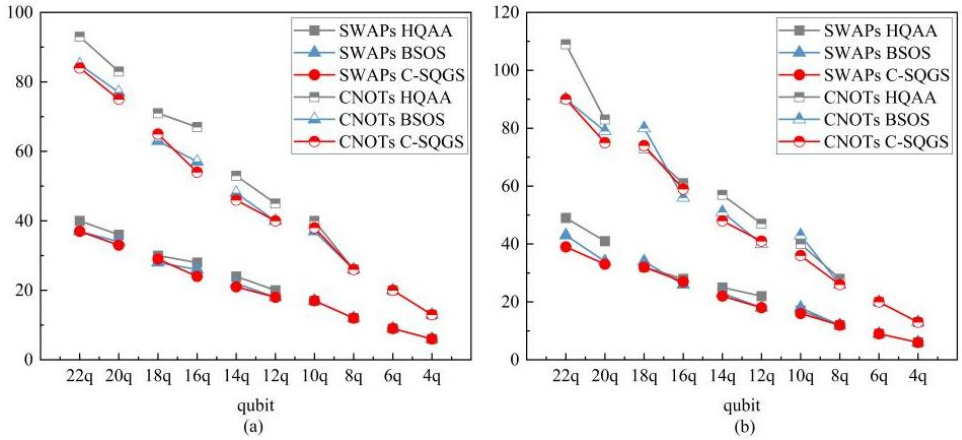

Fig. 8. Comparison of compilation overhead of different optimization algorithms. (a) Comparison of SWAPs and CNOTs overhead in the t|ket⟩. (b) Comparison of SWAPs and CNOTs overhead in the Qiskit.

The research in this paper focuses on the qubit mapping challenge, emphasizing the extra SWAP gate overhead and also considering the change in time complexity, which is a key measure of the algorithm's running efficiency. Table I shows the results of comparing the average running time of the 2QAN with the C-SQGS strategy in the t|ket⟩ and Qiskit compilers. The time complexity increases gradually with the increase in the count of qubits. Specifically, for smaller-scale quantum circuits, such as the 4-qubit model, the running time in the t|ket⟩ and Qiskit compilers is about 0.01s and 0.07s, respectively. However, the running time increases significantly as the circuit size increases. As an example, the runtime in the t|ket⟩ and Qiskit compilers for a 22-qubit circuit is about 27.4s and 0.29s, respectively.

TABLE I. COMPARE THE AVERAGE RUNTIME OF 2QAN AND C-SQGS STRATEGY IN T|KRT⟩ AND QISKIT COMPILERS

Qubit	Compiler	Running time(t/s)	
		2QAN	C-SQGS
4	t|ket⟩	0.1	0.01
	Qiskit	0.01	0.07
6	t|ket⟩	0.3	0.28
	Qiskit	0.09	0.09
8	t|ket⟩	0.46	0.41
	Qiskit	0.1	0.09
20	t|ket⟩	11.79	11.18
	Qiskit	0.24	0.22
22	t|ket⟩	28.23	27.4
	Qiskit	0.30	0.29

V. SUMMARY

This paper provides an in-depth discussion of the qubit mapping challenge, with a particular focus on the impact of physical topology on the mapping process in quantum computing. In the NISQ era, qubit mapping plays a key role in quantum circuit compilation. It aims to efficiently map logical qubits onto physical qubits, minimizing SWAP gates caused by topological constraints and enhancing circuit performance. To address this challenge, the C-SQGS strategy is proposed, aiming to optimize quantum circuits for the 2-local quantum simulation problem. To test the strategy, multiple experiments are conducted. The experimental results demonstrate that the C-SQGS strategy exhibits some advantages in reducing SWAP gate and hardware gate overhead. Specifically, the quantity of SWAP gates is diminished by an average of 36.9% and 47.7%, and the quantity of two-qubit gates is diminished by an average of 13.8% and 13.5% on the t|ket⟩ and Qiskit compilers, respectively.

Beyond empirical performance improvements, the results indicate several broader implications for qubit mapping and quantum compilation. Specifically, qubit flexibility–driven routing, centric gate prioritization, and multi-factor SWAP evaluation emerge as key design elements that help avoid short-sighted routing decisions and enable more execution-aware optimization under hardware constraints.

Although the C-SQGS strategy has achieved certain results in reducing the number of SWAP gates and the overhead of two-qubit gates, it still has limitations. First, this method is primarily designed and experimentally verified for medium-scale quantum circuits. For larger-scale quantum circuits, the computation of its multi-factor cost function and the evaluation process of candidate SWAP sets may introduce significant computational overhead, and the scalability of the algorithm requires further investigation. Second, experimental evaluations are primarily based on typical quantum circuits and a limited range of quantum chip topologies. Its applicability and generalization capabilities in more complex application scenarios and diverse quantum algorithms still have room for improvement.

With the continuous advancement of quantum computing technology, qubit mapping — a central problem in quantum computing — still faces significant challenges. Future research needs to explore several directions in greater depth to meet the evolving demands of quantum hardware and algorithms. On one hand, based on the C-SQGS strategy, optimization algorithms capable of adapting to more complex quantum topologies and higher-dimensional quantum gate sets will be developed, further enhancing the versatility and adaptability of the approach. On the other hand, efforts will be devoted to

designing more scalable mapping algorithms that can handle larger quantum circuits, while simultaneously pursuing the joint optimization of circuit design and mapping strategies. Since qubit mapping is closely linked to circuit structure, optimizing their interplay can further reduce redundant operations and improve the overall efficiency of quantum computing, thereby laying a solid foundation for its practical applications.

ACKNOWLEDGMENT

This work was supported by the Natural Science Foundation of Heilongjiang Province of China [Grant Number: LH2024F042].

REFERENCES

[1] Zhang X, Zhang F. Variational quantum computation integer factorization algorithm[J]. International Journal of Theoretical Physics, 2023, 62(11): 245.

[2] Au-Yeung R, Chancellor N, Halffmann P. NP-hard but no longer hard to solve? Using quantum computing to tackle optimization problems[J]. Frontiers in Quantum Science and Technology, 2023, 2: 1128576.

[3] Paudel H P, Syamlal M, Crawford S E, et al. Quantum computing and simulations for energy applications: Review and perspective[J]. ACS Engineering Au, 2022, 2(3): 151-196.

[4] Niu S, Suau A, Staffelbach G, et al. A hardware-aware heuristic for the qubit mapping problem in the nisq era[J]. IEEE Transactions on Quantum Engineering, 2020, 1: 1-14.

[5] Itoko T, Raymond R, Imamichi T, et al. Optimization of quantum circuit mapping using gate transformation and commutation[J]. Integration, 2020, 70: 43-50.

[6] Wille R, Burgholzer L. MQT QMAP: Efficient quantum circuit mapping[C]//Proceedings of the 2023 International Symposium on Physical Design. 2023: 198-204.

[7] Liu H, Zhang B, Zhu Y, et al. QM-DLA: an efficient qubit mapping method based on dynamic look-ahead strategy[J]. Scientific Reports, 2024, 14(1): 13118.

[8] Li H, Lu K, Qin H, et al. Research on Qubit Mapping Technique Based on Batch SWAP Optimization[J]. International Journal of Advanced Computer Science & Applications, 2023, 14(12).

[9] Chatterjee Y, Bourreau E, Rančić M J. Solving various NP-hard problems using exponentially fewer qubits on a quantum computer[J]. Physical Review A, 2024, 109(5): 052441..

[10] Luteberget B, Pettersen K F, Sartor G, et al. An Exact Branch and Bound Algorithm for the generalized Qubit Mapping Problem[J]. arXiv preprint arXiv:2508.21718, 2025.

[11] Siraichi M Y, Santos V F, Collange C, et al. Qubit allocation[C]//Proceedings of the 2018 international symposium on code generation and optimization. 2018: 113-125.

[12] Zulehner A, Paler A, Wille R. An efficient methodology for mapping quantum circuits to the IBM QX architectures[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, 38(7): 1226-1236.

[13] Steinberg M A, Feld S, Almudever C G, et al. Topological-graph dependencies and scaling properties of a heuristic qubit-assignment algorithm[J]. IEEE Transactions on Quantum Engineering, 2022, 3: 1-14.

[14] Li Z, Liu P, Zhao P, et al. Error per single-qubit gate below 10^{-4} in a superconducting qubit[J]. npj Quantum Information, 2023, 9(1): 111.

[15] Lau J W Z, Haug T, Kwek L C, et al. NISQ Algorithm for Hamiltonian simulation via truncated Taylor series[J]. SciPost Physics, 2022, 12(4): 122.

[16] Faehrmann P K, Steudtner M, Kueng R, et al. Randomizing multi-product formulas for Hamiltonian simulation[J]. Quantum, 2022, 6: 806.

[17] Zhang Z J, Sun J, Yuan X, et al. Low-depth Hamiltonian simulation by an adaptive product formula[J]. Physical Review Letters, 2023, 130(4): 040601.

[18] Dong Y, Lin L, Ni H, et al. Infinite quantum signal processing[J]. Quantum, 2024, 8: 1558.

[19] Motlagh D, Wiebe N. Generalized quantum signal processing[J]. PRX Quantum, 2024, 5(2): 020368.

[20] Childs A M, Maslov D, Nam Y, et al. Toward the first quantum simulation with quantum speedup[J]. Proceedings of the National Academy of Sciences, 2018, 115(38): 9456-9461.

[21] Khumalo M. Review of Quantum Optimisation Techniques for the Quadratic Assignment Problem[J]. 2023.

[22] Li G, Ding Y, Xie Y. Tackling the qubit mapping problem for NISQ-era quantum devices[C]//Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems. 2019: 1001-1014.

[23] Lao L, Browne D E. 2qan: A quantum compiler for 2-local qubit hamiltonian simulation algorithms[C]//Proceedings of the 49th Annual International Symposium on Computer Architecture. 2022: 351-365.

[24] Niemann P, Mueller L, Drechsler R. Combining SWAPs and remote CNOT gates for quantum circuit transformation[C]//2021 24th Euromicro Conference on Digital System Design (DSD). IEEE, 2021: 495-501.

[25] Deng H, Zhang Y, Li Q. Codar: A contextual duration-aware qubit mapping for various nisq devices[C]//2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020: 1-6.

[26] Lao L, van Wee B, Ashraf I, et al. Mapping of lattice surgery-based quantum circuits on surface code architectures[J]. Quantum Science and Technology, 2018, 4(1): 015005.

[27] Tannu S S, Qureshi M K. A case for variability-aware policies for nisq-era quantum computers[J]. arXiv preprint arXiv:1805.10224, 2018.

[28] Sivarajah S, Dilkes S, Cowtan A, et al. t|ket⟩: a retargetable compiler for NISQ devices[J]. Quantum Science and Technology, 2020, 6(1): 014003.

[29] Neha K. Quantum programming: working with IBM'S qiskit tool[J]. The Scientific Temper, 2023, 14(01): 93-99.