Research on Qubit Mapping Technique Based on Batch SWAP Optimization

Hui Li, Kai Lu, Zi'ao Han, Huiping Qin, Mingmei Ju, Shujuan Liu
School of Computer and Information Engineering, Harbin University of Commerce, Harbin, China

***Abstract*—The conventional approach for initial qubit mapping in the Noisy Intermediate-Scale Quantum (NISQ) era typically uses a static heuristic strategy, overlooking insufficient qubit neighborhood in subsequent operations, resulting in excess additional SWAP gates. To address this, we introduce a multifactor interaction cost function considering qubit distance, interaction time, and gate operation error rates, enhancing SWAP gate selection in the traditional strategy. Considering quantum hardware constraints, we propose Batch SWAP Optimization Strategy (BSOS). BSOS tackles qubit mapping challenges by leveraging optimal SWAP gate selection and a SWAP-based batch update technique, effectively minimizing SWAP gates throughout circuit execution. Experimental results show that BSOS significantly reduces additional gates by intelligently selecting SWAP gates and using batch updating, with a 38.1% average decrease in inserted SWAP gates, leading to a 12% reduction in hardware gate counting overhead.**

***Keywords*—Quantum computing; quantum circuit compilation; initial qubit mapping; Batch SWAP Optimization Strategy (BSOS); best SWAP choice; Batch Update Technology (BUT)**

I. INTRODUCTION

Quantum computing, a revolutionary paradigm, has transformed finance [1], machine learning [2], optimization [3], and chemistry [4]. Traditional computers face challenges in complex problem solving and large-scale data processing due to resource limitations. Quantum computers offer a new solution with inherent parallelism. In quantum computing, Hamiltonian quantities describe problem evolution, translated for simulation using algorithms like Grover's search [5] and Shor's algorithm [6]. The product formula, approximating Hamiltonian exponentiation, is crucial. Quantum circuits excel in manipulating exponents, making them powerful for Hamiltonian simulation. Leveraging quantum circuit advantages efficiently advances quantum computing.

High-level representations of quantum circuits are not inherently tied to specific hardware and require translation into an instruction set compatible with the underlying quantum hardware for execution. In NISQ computers, the prevalent instruction set typically comprises a single-qubit rotation gate and one or more two-qubit gates. These quantum computers can only apply two-qubit gates to a limited set of qubit pairs due to constrained connections between qubits. Consequently, it becomes essential to perform circuit-to-hardware qubit mapping using initial qubit mapping techniques. Additionally, to scale up the circuit by increasing the number of gates and depth, it is necessary to reposition qubits to neighboring locations by introducing SWAP gates.

The existing literature on time scheduling strategies [7] offers solutions for compiling circuit depth, yet it may encounter challenges in managing constraints and optimization, particularly with intricate circuits. Another approach in [8], focusing on time scheduling and constraint planning, generates a hot-start solution but may face limitations regarding computational complexity and algorithmic efficiency, especially for large-scale circuits. A proposed greedy stochastic search approach [9] proves effective for similar problems but may struggle with complex circuits and global optimization. Genetic algorithms with chromosome coding strategies, introduced in [10] for optimization, might face challenges related to algorithmic parameter sensitivity and convergence speed. The study in [11] explores the trade-off between switched gates and circuit depth during compilation but offers limited consideration of hardware characteristics such as gate error rate. The research in [12] and [13] introduces strategies considering gate error rates, a significant advancement, yet practical applications may necessitate a more comprehensive consideration of hardware characteristics like cooling time and connectivity. In summary, while these literatures contribute valuable insights to the problem, further improvements and a more holistic approach may be required to address complex circuits and global optimization effectively.

In this paper, we explore Hamiltonian operator arrangements' flexibility, propose a multifactor interaction cost function and introduce BSOS for the initial qubit mapping process, aiming to efficiently compile quantum circuits for 2-local qubit Hamiltonian simulation problems. Given prevalent NISQ computer characteristics, where two-qubit gate error rates are typically 10 times higher than single-qubit gates, and qubit coherence time is shorter [14], BSOS adapts to diverse qubit topologies and gate sets. It seamlessly integrates with various quantum circuit mapping algorithms and is suitable for quantum approximation optimization algorithms like Quantum Approximate Optimization Algorithm (QAOA) [15]. Through evaluations, BSOS substantially reduces the required gate number compared to the state-of-the-art initial qubit mapping strategy. We further validate the effectiveness of BSOS through experiments on an IBM quantum device.

The main contributions of this paper can be summarized as follows:

- We focus on the initial qubit mapping phase, identifying the challenges and limitations that are the primary focus of this study.
- We introduce a multifactor interaction cost function, considering qubit distance, interaction time, and gate

error rate. This cost function facilitates a more comprehensive evaluation and optimization of quantum circuit performance.

- We propose an optimal SWAP gate selection algorithm and define SWAP gains. The SWAP gain is assessed based on the benefits obtained by adding SWAP gates. Optimal SWAP gates are selected for different instruction sets in quantum circuits.
- We design a scalable SWAP-based batch update technique, providing comparable results to previous mapping-based update-by-sequence approaches. This rapid update scheme ensures the scalability of qubit mapping, allowing the BSOS to adapt to various hardware architectures and accommodate larger quantum devices in NISQ era.

The subsequent sections of the paper are organized as follows: Section II presents relevant background information on quantum simulation and quantum circuits. Section III introduces the modification scheme for the cost function and details the BSOS strategy. The evaluation of these approaches is discussed in Section IV. Finally, the paper is summarized in Section V.

II. PRELIMINARIES

A. Product Formula (Trotter's Formula)

In the realm of quantum computing, the product formula, also known as Trotter's formula, stands out as a fundamental technique for constructing efficient circuit structures. It leverages the decomposition of Hamiltonian quantities, representing the time evolution of a system in exponential form. The essence of the product formula lies in its ability to approximate time evolution by breaking down the system's Hamiltonian quantity, denoted as H, into distinct operators comprising sums of polynomial ergodic terms. Quantum circuits are then employed to efficiently implement these operators. The formula can be succinctly expressed as:

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

where, $V(t)$ represents the state of the system at moment t, L is the number of terms, h_j is the coefficient of the *jth* term and H_j is the corresponding ergodic operator.

If all terms are exchangeable (i.e., $H_jH_k = H_kH_j$), then the product formula approximates the true time evolution U, (i.e., $V(t) = U$). However, in natural physical systems, non-exchangeable terms are typically present. In such cases, a first-order approximation can be used to approximate $V(t)$ as an rth power of $V(t/r)$, where r is a constant greater than 1. This process is known as the Trotterization step, and repeating this step r times forms a Trotter sequence. By decreasing the value of r, the cost of the simulation can be significantly reduced.

In this paper, we mainly consider the 2-local qubit Hamiltonian quantity, as shown in Eq. (2):

$$H = \sum_{(u,v)\in E} H_{uv} + \sum_{k\in V} H_k \tag{2}$$

where, H_{uv} represents a two-qubit Hamiltonian term and H_k is a single-qubit Hamiltonian. The interaction graph of this Hamiltonian quantity is represented by **G**(**V**, **E**), where **V** represents the set of qubits and **E** represents the set of edges.

B. Quantum Circuit

In quantum computing, data is stored in qubits, each of which has two fundamental states, denoted by $|0\rangle$ and $|1\rangle$. Unlike classical bits, qubits can be in a superposition of these two fundamental states, i.e., $\alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex numbers and satisfy $|\alpha|^2 + |\beta|^2 = 1$.

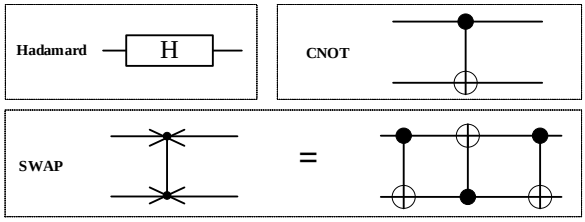

Fig. 1. Basic gates of IBM quantum computers.

Operations in quantum computing are realized through quantum gates, which apply specific operations like rotations, flips, etc., between qubits. Quantum gates are mathematically represented by unitary matrices. A Hadamard gate operates on a single qubit, while both CNOT gates and SWAP gates act on two qubits, as shown in Fig. 1. A CNOT gate flips the state of the target qubit based on the state of the controlling qubit, i.e., CNOT: $|c\rangle|t\rangle \rightarrow |c\rangle|c \oplus t\rangle$, where $c, t \in \{0, 1\}$ and $\oplus$ denotes a heteroskedastic operation. the SWAP gate then exchanges the states of the two target qubits: for all $a, b \in \{0, 1\}$, it will $|a\rangle|b\rangle \rightarrow |b\rangle|a\rangle$. The SWAP gate can be realized by a combination of 3 CNOT gates (see Fig. 1).

A quantum circuit is a framework for describing and manipulating quantum information, comprising a sequence of quantum gates akin to classical logic gates. Quantum gates enact transformations on quantum states. Quantum circuits can be conceptualized as systems constructed from a combination of fundamental quantum gates and quantum measurements. In the realm of intricate quantum operations, like simulating Hamiltonian quantities, quantum gates within quantum circuits can be deliberately designed and fine-tuned.

The quantum mapping task involves a graph **G = (V, E)** that represents the structure of the target quantum device and a circuit **C** representing an ideal quantum algorithm. The gates in circuit **C** are decomposed into elementary gates supported by the target quantum device. The objective of quantum mapping is to transform circuit **C** into a functionally equivalent circuit. In this transformed circuit, each two-qubit gate acts on a pair of neighboring nodes in the graph **G** of the target quantum device. Essentially, the goal is to adapt and map the circuit **C** according to the physical architecture of the target quantum device, ensuring its compatibility with the specific quantum hardware. Fig. 2(a) shows an example of a circuit where **Q** = $\{q_0, ..., q_4\}$, **C** = $\{g_0, ..., g_6\}$, where g_0 = CNOT (q_0, q_1), g_1 = CNOT (q_2, q_4), and so on. In the figure, each CNOT gate is labelled with the qubit they act on. In the text, *q is* used to represent logical qubits and *P* is used to represent physical qubits.

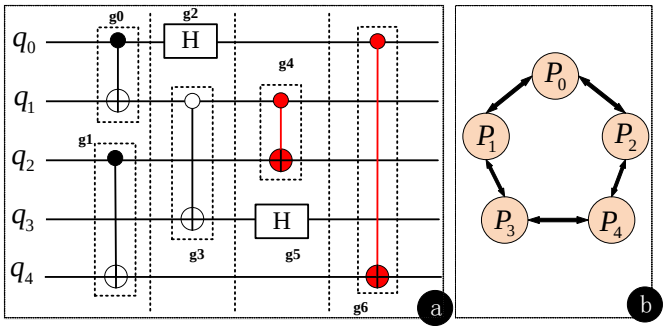

Fig. 2. (a) A logical circuit with depth 4, (b) Architecture diagram of quantum device.

Circuit **C** is typically represented as a sequence of gates (g_0, g_1, ..., g_{m-1}), but this cannot imply that in all cases the (i + s)th gate has to be executed after the *ith* gate (where $i \geq 0$, $s \geq 1$, and $0 < i + s < m$). In reality, if the two gates don't involve common qubits, they can be executed in parallel. For instance, considering the circuit in Fig. 2(a), it can be expressed as:

$$C = \left(\langle q_0, q_1\rangle, \langle q_2, q_4\rangle, \langle q_0\rangle, \langle q_1, q_3\rangle, \langle q_1, q_2\rangle, \langle q_3\rangle, \langle q_0, q_4\rangle\right).$$

III. Proposed Methodologies

A. Problem in Initial Mapping

Qubit mapping aims to determine the optimal arrangement of qubits, minimizing the number of qubit shift operations needed for all two-qubit gates. Like the methodologies outlined in [16]-[19], the qubit mapping problem is cast as a Quadratic Assignment Problem (QAP).

Similar to the previous approach, a SWAP operation is employed to alter the state between two qubits, facilitating the adjustment of qubit mapping. Introducing 1 SWAP gate increases the circuit's depth by 3. Multiple swap gates enable the relocation of a logical qubit to any physical qubit position. Fig. 3 illustrates that after inserting a SWAP operation between q_0 and q_2 following the third CNOT gate, the modified quantum circuit becomes executable. The first 3 CNOT gates can be executed under the initial qubit mapping, and after inserting the SWAP, the mapping is updated to $\{q_0 \rightarrow p_2, q_1 \rightarrow p_1, q_2 \rightarrow p_0, q_3 \rightarrow p_3, q_4 \rightarrow p_4\}$, and the remaining two CNOT gates can now be executed under this updated mapping.

Comparing the initial and optimized circuits in Fig. 2(a) and Fig. 3, it is evident that the number of gates increases from 7 to 10, and the circuit depth increases from 4 to 7. The introduction of additional SWAP gates notably enhances the circuit's execution time. Hence, the primary objective of the mapping process is to minimize the number of additional SWAP gates inserted, aiming to reduce the overall error rate and total execution time of the final hardware-adapted circuit.

Definition 1 Qubit mapping [6]: given a coupling map of input quantum circuits and quantum devices, find the initial qubit mapping and the intermediate qubit mapping transformations (by insertion swapping) to satisfy all two qubit constraints and try to minimize the number of additional gates and the circuit depth in the final hardware-compatible circuit.

In the initial qubit mapping phase of quantum circuit compilation, several challenges need to be addressed. This paper considers the following limitations:

- Selection of physical qubits: It is crucial to choose the appropriate physical qubits to map the logical qubits. Taking into account the hardware's topology, physical qubits are selected to minimize communication overhead.
- Connectivity limitation: Some hardware platforms only permit direct communication between specific qubits, while other communications need to be achieved through SWAP gates.
- SWAP gate cost: The execution cost of SWAP gates is typically high, so the cost of SWAP gates is taken into account when choosing the initial qubit mapping scheme, with a preference for less costly SWAP operations.
- Optimal performance trade-off: The goal of the initial qubit mapping is to achieve efficient quantum gate operations while minimizing the communication overhead. Different performance metrics such as communication overhead, SWAP gate cost, etc., need to be weighed when choosing the physical qubits and order.

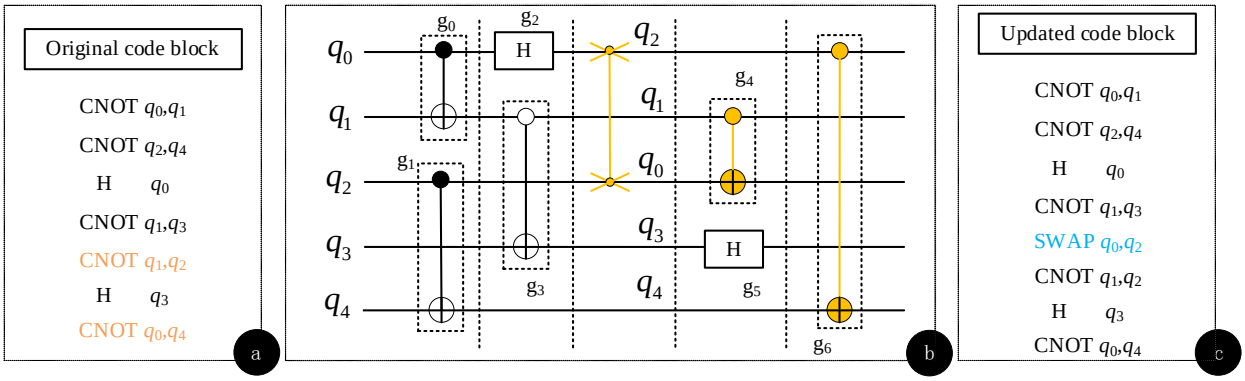

Fig. 3. (a) Original code block, (b) Updated hardware-compliant quantum circuit, (c) Updated code block.

Improving the quality of the initial qubit mapping requires addressing the following questions:

- How to determine the appropriate mapping targets? This involves identifying, for a selected gate, which qubits are the targets. In other words, it determines which qubits need to be communicated or exchanged during the execution of the gate. Once the best mapping target has been selected, the appropriate mapping operation can be performed, such as the execution of a SWAP gate to exchange the positions of qubits. This ensures that the target operation can be executed correctly in hardware.
- How to efficiently choose where to insert SWAP gates, and how many to insert, in order to improve the mapping and minimize the total number of SWAP gates.

B. Design the Cost Function

In the exploration of potential SWAP gates, the role of the cost function is to prioritize the SWAP gate that closely aligns with the target mapping in terms of cost. Assessing potential SWAP gates entails the selection of the most favorable SWAP sequence, constituting an ongoing decision-making process integral to the overall mapping strategy. The arrangement requires reevaluation when updating the qubit mapping following each SWAP operation, constituting a dynamic process. To achieve optimal mapping outcomes, the mapping must be recalibrated and adjusted subsequent to each SWAP insertion through the utilization of a cost function.

Among numerous mapping methods, the cost function serves as a pivotal tool for assessing the effectiveness of a mapping strategy. However, prevailing cost functions typically focus on a singular factor, predominantly relying on distance as the fundamental metric—whether physical or logical distance between qubits. These functions often lack the capability to consider multiple factors. Physical distance impacts the operational speed and error rate, as qubits situated farther apart may necessitate more steps and involve intermediate bits with a higher probability of errors during operations.

To address the aforementioned issue, this paper introduces a multifactor interaction cost function as shown in Eq. (3). This function incorporates three primary factors: distance, interaction time, and error rate of gate operation. Minimizing the cost function enables the identification of the most efficient and accurate strategies and configurations for accomplishing a specific quantum computing task. This cost function is

$$Y_{cost}(E, f_{ij}, d_{\phi(i)\phi(j)}) = \min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} E f_{ij} (d_{\phi(i)\phi(j)})^2 \quad (3)$$

where, S_n denotes all possible permutations, f_{ij} represents the interaction time between logic qubits i and j in the circuit, and the interaction time can be estimated based on hardware specifications or empirical experiments. e denotes the probability of an error occurring for the execution of a double quantum gate, where $0 \leq E \leq 1$, and $d_{\varphi(i)\varphi(j)}$ denotes the physical distance between the qubits $\varphi(i)$ and $\varphi(j)$ on the hardware, which can be calculated by using the Floyd-Warshall algorithm to perform the calculation.

The physical distance between qubits plays a pivotal role in influencing the speed and efficacy of their interactions. Increased distances directly contribute to heightened computational complexity and error rates, with the square factor of distance exerting a substantial impact on overall computational cost, thereby influencing the efficiency and accuracy of quantum computation. Interaction time serves as a determinant of task execution speed, with prolonged interaction times resulting in extended task execution durations. Introducing interaction time as a factor in the cost function underscores its significance in determining the overall computational cost and efficiency. Quantum gate operations inherently incur error rates, where elevated error rates undermine computational accuracy and reliability. The incorporation of gate operation error rates into the cost function underscores the critical influence of accuracy and reliability on the cost and efficiency of quantum computing. By amalgamating these three factors, the initial qubit mapping of a quantum circuit can be generated with a more comprehensive consideration of hardware limitations and practical implementation constraints.

C. Best SWAP Choice

In the realm of quantum computing, any multi-quantum gate can be decomposed into a combination of single quantum gates and CNOT gates. The execution of these gates necessitates the inclusion of SWAP gates to alter the connectivity pattern between qubits. To minimize the surplus of added SWAP gates, strategic selection of SWAP gates that can yield more nearest-neighbor (NN) gates is crucial. This practice contributes to the implementation of intricate quantum operations and algorithms, enhancing the coherence of the circuit. The significance of SWAP gates lies in two primary aspects: Firstly, SWAP gates facilitate the interchange of states between two qubits, thereby reshaping the relationships between qubits and optimizing the structure and efficiency of the quantum circuit. Secondly, in practical quantum operations, achieving the target superposition quantum state often involves constructing a combination of corresponding quantum gates. A higher count of NN gates enhances the likelihood of attaining this objective. Formally, SWAP gate gains are defined as follows:

Definition 2 SWAP gate gains: let P be the set of non-nearest-neighbor (non-NN) gates, Q be the set of executable SWAP gates, V_{SWAP} = **{P, Q}** be the list of SWAP gains, N_1 be the set of executable $\mathbf{V_{SWAP}}$, i.e., the number of NN gates after adding this SWAP gate. N_0 is the number of NN gates when the $\mathbf{V_{SWAP}}$ set is not executed. The optimal benefit $\mathbf{V_{SWAP}}$ set is determined by selecting max $\{N_1 - N_0\}$.

$$V_{SWAP} = N_1 - N_0 \quad (4)$$

We introduce a heuristic-based strategy designed to optimize the selection of SWAP gates, with the goal of minimizing the compilation overhead in quantum circuits. The strategy involves simulating the execution of a SWAP operation for each potential candidate, resulting in a new mapping. Subsequently, the sizes of the elements in VSWAP

are compared, determined by the number of NN gates generated under the new mapping with different SWAP gates inserted. Among all feasible SWAP operations, the one with the largest element size is chosen. This approach enables a more intelligent selection of SWAP gates, thereby reducing the overall count of SWAP gates and enhancing the overall performance of the quantum circuit.

Algorithm 1 Best SWAP Choice

Input.
nn_gate_count: The number of nearest-neighbor gates in the circuit after adding a swap gate between two qubits
moves: A set of insertable SWAP gates with the same cost
Output.
best_move: Optimal insertable SWAP gate

```
begin
    max_nn_gate_increase ← 0
    best_move ← None
    for move in moves do
        nn_gate_increase ← nn_gate_count[move]
            if nn_gate_increase > max_nn_gate_increase
                max_nn_gate_increase ← nn_gate_increase
                best_move ← move
        end if
    end for
end
```

Details of the pseudo-code can be found in Algorithm 1 in the text, while a specific application example is provided in Fig. 4.

Compile an 8-qubit 2-local Hamiltonian into the lattice architecture depicted in Fig. 4. In the presence of a set of insertable SWAP gates with identical costs, the approach outlined in this paper is to identify the one with the highest gain among these gates. Fig. 4(a) presents a qubit map of a circuit along with a scenario where specific quantum operations algorithms cannot be implemented without the insertion of SWAP gates. The upper figure displays the inserted SWAP gates and the CNOT gates implemented after insertion, while the lower figure illustrates the qubit graph, where nodes represent qubits, and edges signify their connectivity. To prevent confusion, the SWAP gates in the figure are applied to the corresponding hardware qubits. For enhanced readability, they are plotted on the circuit qubits.

In the circuit, CNOT (0, 1) and CNOT (1, 5) are direct mappings, but SWAP gates are needed to perform the remaining CNOT gates. SWAP gates with the same cost are calculated according to Eq. (3), such as SWAP (2, 5) and SWAP (5, 6) in figure. Subsequently, the filtered SWAP gates are inserted into the circuits separately to see the number of NN gates generated under the new mapping. As in Fig. 4(b) SWAP (q_2, q_5) is inserted and V_{SWAP} = 2 under the current mapping (see Eq. 4). Whereas in Fig. 4(c), V_{SWAP} = 1 after inserting SWAP (q_5, q_6) By comparison, a set of SWAP gates with a larger number of NN gates is selected and inserted into the circuit to update the mapping. And so on until all the quantum gates are mapped and the final mapping result is shown in Fig. 4(d). This strategy ensures that the selected SWAP gates maximize the number of NN gates and optimize the mapping of the quantum circuit.

D. Batch Update Technology

The traditional approach to quantum mapping involves real-time updates of mappings to adjust the relationship after each SWAP operation. However, as quantum circuits increase in size, this method becomes inefficient. To address this issue, we propose a Batched Update Technique (BUT).

The design of the BUT is based on reducing the cost associated with frequent mapping updates in conventional quantum mapping methods. The strategy aims to enhance the efficiency of quantum circuit mapping by decreasing the frequency of updates, incorporating the overall circuit structure, and balancing mapping performance with the associated update costs, among other theoretical considerations. The fundamental concept is to holistically consider multiple mapping update operations to reduce interference with circuit execution operations and achieve a trade-off between mapping quality and update costs. Specifically, each batch update takes into account all relevant SWAP operations and executes mapping updates for these operations simultaneously. This approach reduces the number of mapping updates stemming from a single SWAP operation, thereby diminishing latency and energy consumption in quantum computing.

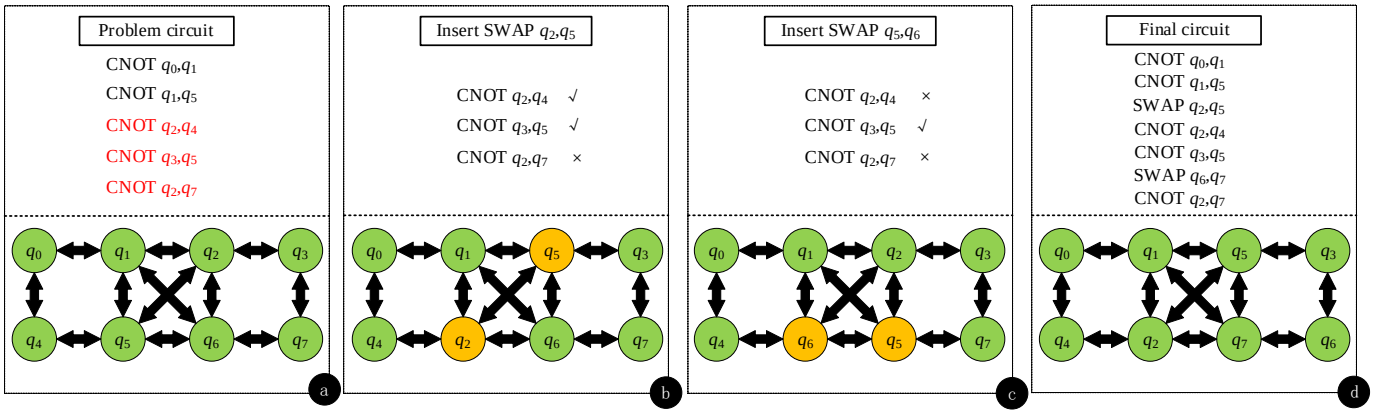

Fig. 4. Examples of compiling a 8-qubit 2-local Hamiltonian to a grid architecture. (a) A problem circuit, (b-c) Insert SWAP. (b) The NN gate that can be realized after inserting SWAP (q_2, q_5), (c) The NN gate that can be realized after inserting SWAP (q_5, q_6), (d) Final circuit.

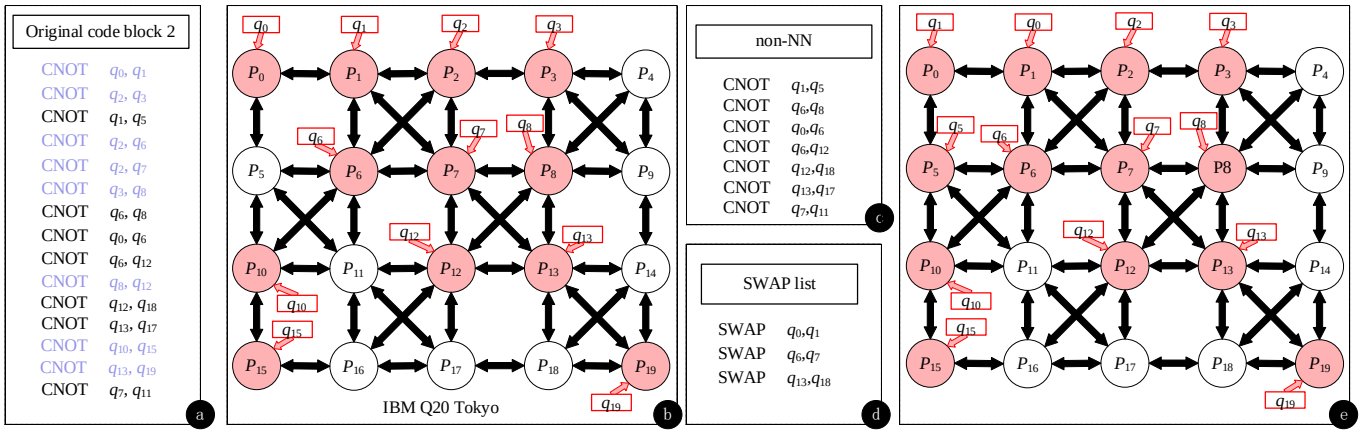

Fig. 5. (a) Original code block 2, (b) Coupling graph of IBM q20, (c) non-NN quantum gate, (d) List of SWAP gates to be executed, (e) Batch update mapping results.

In extensive quantum circuits, the interaction constraints between qubits necessitate multiple SWAP operations to facilitate the exchange between non-adjacent qubits. Each SWAP operation triggers a mapping update, contributing to inefficiency. Consequently, consolidating multiple operations and updating the mapping collectively after completing all operations emerges as an appealing solution. The BUT scrutinizes the entire quantum circuit to identify which two-qubit gates can be optimized through a shared SWAP operation. When a SWAP gate is chosen for execution, it gets added to a "list of SWAP operations to be performed," with a defined condition dictating when to cease additions, such as reaching a predetermined list length. During batch execution of SWAP operations, the algorithm iterates through the list, executes all listed SWAP operations, and updates the qubit mapping collectively upon completion.

Fig. 5 shows an example, assuming the circuit shown in Fig. 5(a) is run on the 20-qubits device Tokyo (Fig. 5(b)). First, two-qubit gates that are NNs on the initial qubit layout, such as those labelled purple in Fig. 5(a), are filtered from the original circuit list2. Map these gates directly to the corresponding hardware (e.g., the qubits labelled pink in Fig. 5(b)), i.e.

$q_0 \rightarrow p_0$, $q_1 \rightarrow p_1$, $q_2 \rightarrow p_2$, $q_3 \rightarrow p_3$, $q_6 \rightarrow p_6$, $q_7 \rightarrow p_7$, $q_8 \rightarrow p_8$, $q_{10} \rightarrow p_{10}$, $q_{12} \rightarrow p_{12}$, $q_{13} \rightarrow p_{13}$, $q_{15} \rightarrow p_{15}$, $q_{19} \rightarrow p_{19}$.

For non-NN two-qubit gates, the costs between qubits are compared by means of a computed cost function. The qubit pair with the smallest cost is chosen as the target of the mapping, assuming that the SWAP gate with the smallest cost is evaluated as $\{q_0, q_1\}$. The previous method is to add that SWAP gate to the execution list, update the qubit mapping, and remove the NN gates from the unmapped set of gates. The steps are repeated until all double qubit gates are mapped. Instead of executing the SWAP gates directly, the strategy in this paper puts the SWAP (q_0, q_1) into the SWAP list (e.g., Fig. 5(d)), and then searches for the next SWAP gate, which is also added to the list. When the length of the list reaches a predefined value, it traverses the "list of SWAP operations to be performed" and then performs all SWAP operations in the list. After all the above SWAP operations are completed, the qubit mapping is updated uniformly.

In this process, handling multiple operations simultaneously may introduce some complexity, as it is necessary to ensure that there are no conflicts between batch operations and to update the qubit mapping correctly. Whenever a new SWAP operation is added to the "pending SWAP operation list," a conflict check is performed for this operation with other operations already in the list. Specifically, it is ensured that the new SWAP operation does not impact or be impacted by the SWAP operations already in the list. The qubit mapping is then updated in bulk based on the selected SWAP list, i.e., when each new SWAP operation is added to the "pending SWAP operation list.", i.e.

$q_0 \rightarrow p_1$, $q_1 \rightarrow p_0$, $q_2 \rightarrow p_2$, $q_3 \rightarrow p_3$, $q_5 \rightarrow p_5$, $q_6 \rightarrow p_7$, $q_7 \rightarrow p_6$, $q_8 \rightarrow p_8$, $q_{10} \rightarrow p_{10}$, $q_{11} \rightarrow p_{11}$, $q_{12} \rightarrow p_{12}$, $q_{13} \rightarrow p_{18}$, $q_{15} \rightarrow p_{15}$, $q_{17} \rightarrow p_{17}$, $q_{18} \rightarrow p_{13}$, and $q_{19} \rightarrow p_{14}$

Remove the NN from the unmapped gates and empty the "list of pending SWAP operations" in preparation for the next batch of SWAP operations. Repeat until all double-qubit gates are mapped.

IV. Result and Discussion

Similar to earlier studies, this paper uses the following metrics to evaluate the performance of different compilers: the aggregate count of inserted SWAP gates (lower values are preferable) and the overall count of executed two-qubit gates on the hardware (lower values are preferable). These metrics enable the evaluation of algorithm performance, particularly in handling intricate circuits. The benchmarking procedure aligns with IBM's Qiskit quantum program, utilizing the Qiskit compiler for decomposition and optimization of the CX/CNOT gate set. The benchmarking methodology outlined in [13], focusing on the QAOA model, was adopted, wherein the time evolution of Hamiltonian quantities is conducted by multiplying Eq. (5) by:

$$V(t) = (\Pi_{j=1}^{L} \exp(ih_j H_j t / r))^r \quad (5)$$

where, r is the number of iterations of Trotter. The coefficients of H_j were randomly selected in the range (0, π). The evaluation ranges from 4 to 22 qubits, running their mapping process five times and selecting the best result.

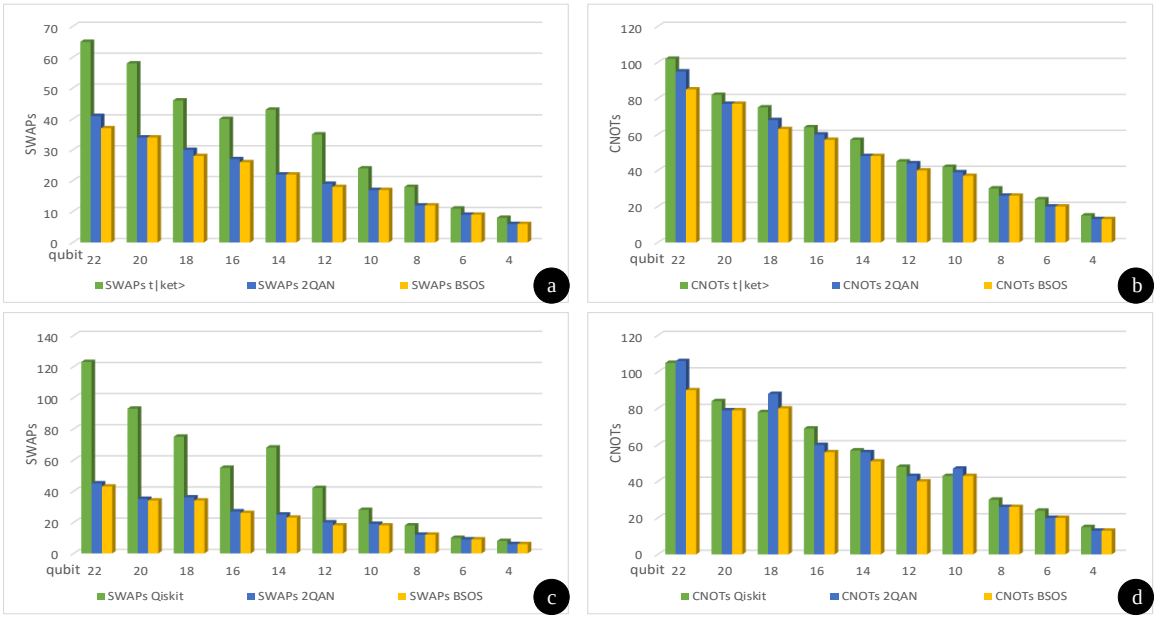

Fig. 6. (a) Comparison of SWAP gate compilation cost between BSOS technology and t|ket⟩ and 2QAN, (b) Comparison of CNOT gate compilation cost between BSOS technology, t|ket⟩ and 2QAN, (c) Comparison between BSOS technology and Qiskit and Comparison of SWAP gate compilation cost of 2QAN, (d) Comparison of BSOS technology, t|ket⟩ and 2QAN, (e) Comparison of SWAP gate compilation cost of 2QAN (c) Comparison between BSOS technology and Qiskit and Comparison of SWAP gate compilation cost of 2QAN, (d) Comparison of BSOS technology with CNOT gate compilation cost of Qiskit and 2QAN.

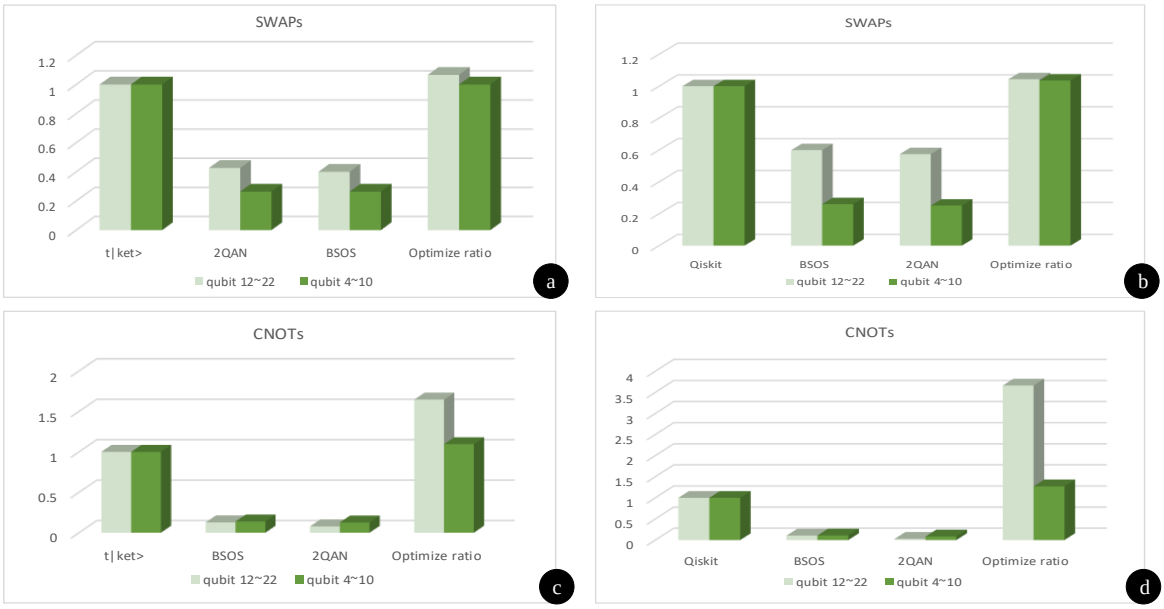

Fig. 7. The BSOS strategy was evaluated for circuits with 4 to 10 qubits and 12 to 22 qubits. (a) SWAP gate optimization rate of BSOS relative to t|ket⟩ and 2QAN, (b) SWAP gate optimization rate of BSOS relative to Qiskit and 2QAN, (c) CNOT gate optimization rate of BSOS relative to t|ket⟩ and (d) CNOT gate optimization rate of BSOS relative to t|ket⟩ and 2QAN. optimization rate of BSOS relative to Qiskit and 2QANs

BSOS was implemented in Python 3.9 and all compilations were performed on a laptop with an Intel Core i5 processor (2.30GHz and 8GB RAM).

Compare the BSOS compilation strategy with the compilation overheads of t|ket⟩ and Qiskit. Fig. 6 and Fig. 7 show the compilation results on the IBM. Compared to t|ket⟩ and Qiskit, BSOS has the least compilation overhead in terms of the number of SWAP gates inserted, the number of hardware dual-quantum gates, and the circuit depth. Specifically, the t|ket⟩ compiler [20] (version 0.11.0) and the Qiskit compiler [21] (version 0.26.2, optimization level 3), equipped with the recommended "FullPass", are considered.

The IBM quantum compiler is limited to CNOT or CZ gate sets, so the models were evaluated using the compilation results on the IBM quantum computer. Of all the benchmarks and quantum computers, the run using the BSOS strategy in the QAP mapping turned out to be the best, as can also be seen in Fig. 6, where the optimization is more pronounced with a

higher number of qubits. For 22 qubits, BSOS inserts 43% less SWAP counts than t|ket⟩ and 65% less than Qiskit (Fig. 6(a) (b)). This reduction in SWAP count will lead to a reduction in the number of hardware double-qubit gates, and BSOS reduces the double-qubit gate overhead by 17% and 14% (Fig. 6(c) (d)).

In the t|ket⟩ compiler, for the case of 4 to 10 qubits, the average reduction in the number of inserted SWAP gates is 26.4%, while the average reduction in the number of inserted CNOT gates is 13.8%. This optimization effect is 10% higher than using only 2QAN. After optimization for 12 to 22 qubits, the average reduction in the number of inserted SWAP gates across all evaluated benchmarks is 40%, and the average reduction in the number of inserted CNOT gates is 12.8%. This optimization effect is 70% higher than using only 2QAN (see Fig. 7(a) (b)).

In the Qiskit compiler, for 4 to 10 qubits, the average reduction in the number of inserted SWAP gates is 26%, while the average reduction in the number of inserted CNOT gates is 10.8%. This optimization effect is 30% higher than using only 2QAN. For 12 to 22 qubits, after optimization, the average reduction in the number of inserted SWAP gates is 60%, and the average reduction in the number of inserted CNOT gates is 10.6% (see Fig. 7(c) (d)).

In this study, optimizing the placement of qubits is taken as the main concern. To solve this problem efficiently, the Tabu search algorithm was chosen. This algorithm is very fast in solving small scale problems, e.g., for the QAOA model with 4 qubits, it takes only about 0.221 seconds in the t|ket⟩ compiler and about 0.004 seconds in the Qiskit compiler, as shown in Table I. However, the processing speed drops significantly when faced with problems of larger size. For example, the QAOA model with 20 qubits takes about 24.218 seconds in the t|ket⟩ compiler and about 0.0176 seconds in the Qiskit compiler.

By applying this cost function, quantum computing researchers and engineers can more accurately quantify the impact of different designs and strategies on the system performance, providing a scientific basis for decision-making and advancing the development of quantum computing technology. In order to verify the accuracy and effectiveness of the proposed cost function, a comparison experiment is conducted in the t|ket⟩ compiler for the distance-only cost function and the cost function proposed in this paper, and the results are shown in Fig. 8. From the figure, it can be seen that the compilation result of the cost function in this paper is better. For 12~22 bits, SWAP is reduced by 28.52% and CONT is reduced by 35.81% on average. While for 4~10 bits, SWAP is reduced by 7.32% and CONT is reduced by 13.74% on average.

TABLE I. COMPARING AVERAGE RUNNING TIMES OF 2QAN AND BUT IN COMPILERS T|KET⟩ AND QISKIT

qubit	Running time		
		BUT	2QAN
22	t|ket⟩	22.218	22.553
	Qiskit	0.0176	0.0184
20	t|ket⟩	10.274	10.954
	Qiskit	0.0178	0.0178
6	t|ket⟩	0.318	0.361
	Qiskit	0.005	0.006
4	t|ket⟩	0.221	0.244
	Qiskit	0.004	0.004

The superior performance of BSOS in the 22-qubit scenario primarily manifests in its intelligent SWAP gate selection strategy, the comprehensive consideration using a multifactor interaction cost function, and the introduction of batch update techniques. These advantages enable BSOS to more effectively optimize the placement of quantum bits in large-scale quantum circuits. Specifically, the intelligent SWAP gate selection and comprehensive consideration of multiple factors enhance the overall mapping performance, while the batch update technique reduces mapping costs. These optimization effects are particularly pronounced in the case of 22 qubits.

The BSOS algorithm not only demonstrates outstanding performance in the 22-qubit scenario but also holds broad potential applications and future research directions. The application areas of BSOS include the compilation and execution of large-scale quantum circuits, especially well-suited for highly optimized quantum tasks. Future research directions may encompass optimizing nested quantum algorithms, adapting the BSOS algorithm to dynamic scenarios, deeper integration of quantum hardware characteristics, and incorporating BSOS into comprehensive quantum compilation automation tools, providing support for the further development of quantum computing technology.

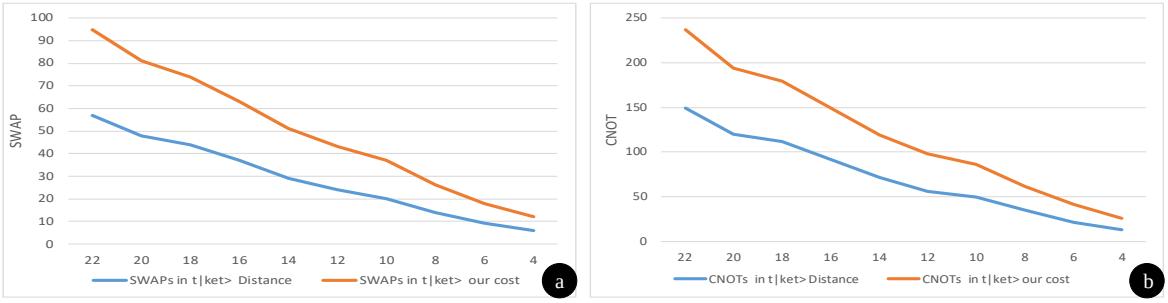

Fig. 8. (a) The number of SWAP gates under different cost functions, (b) The number of CNOT gates under different cost functions.

V. CONCLUSION

In the NISQ era, there is still a significant gap between quantum software and imperfect NISQ hardware. This research introduces a Bulk SWAP Optimization Strategy (BSOS) specifically designed for addressing the 2-local qubit Hamiltonian simulation problem. Focusing on the adaptable operators within Hamiltonian quantities, the primary optimization targets the initial qubit mapping of qubits. A comprehensive evaluation reveals that the BSOS strategy significantly mitigates compilation overhead on the IBM quantum computer, demonstrating superior performance compared to the other two general-purpose compilers.

The optimal SWAP gate selection algorithm optimizes circuit locality by selecting SWAP gates that generate a larger number of newly added NN gates, while the SWAP updating strategy reduces the frequency of mapping by batch updating and optimizing the timing, which improves the overall efficiency of quantum circuit mapping. On the other hand, the introduction of qubit interaction time and the error rate of gate operation in the cost function helps to improve the efficiency and reliability of quantum computation, which makes up for the lack of comprehensiveness and accuracy of previous methods. This makes the proposed method more applicable to NISQ computers with different characteristics and optimization goals, and provides a useful improvement direction for the efficient execution of mesoscale quantum computation. Looking ahead, more optimization work is planned and other possible research directions are explored. By applying error mitigation techniques, it is expected that the error rate can be further reduced, thus further improving the performance and reliability of quantum computation.

ACKNOWLEDGMENT

This work was partially supported by the Natural Science Foundation in Heilongjiang Province of China, under Grant LH2022F035; in part by the University Nursing Program for Young Scholars with Creative Talents in Heilongjiang Province, under Grant UNPYSCT-2020212; and in part by the Science Foundation of Harbin Commerce University, under Grant XL0095.

REFERENCES

[1] Stamatopoulos, N., Egger, D. J., Sun, Y., Zoufal, C., Iten, R., Shen, N., & Woerner, S. (2020). Option pricing using quantum computers. *Quantum*, *4*, 291.

[2] Zoufal, C., Lucchi, A., & Woerner, S. (2019). Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, *5*(1), 103.

[3] Harwood, S., Gambella, C., Trenev, D., Simonetto, A., Bernal, D., & Greenberg, D. (2021). Formulating and solving routing problems on quantum computers. *IEEE Transactions on Quantum Engineering*, *2*, 1-17.

[4] Jang, S. J. (2023). Quantum Mechanics for Chemistry. *Quantum*.

[5] Grover, L. K. (1996, July). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (pp. 212-219).

[6] Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, *41*(2), 303-332.

[7] Cotta, C., & Fernández, A. J. (2007). Memetic algorithms in planning, scheduling, and timetabling. In *Evolutionary Scheduling* (pp. 1-30). Berlin, Heidelberg: Springer Berlin Heidelberg.

[8] Booth, K., Do, M., Beck, J., Rieffel, E., Venturelli, D., & Frank, J. (2018, June). Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. In *Proceedings of the International Conference on Automated Planning and Scheduling* (Vol. 28, pp. 366-374).

[9] Oddi, A., & Rasconi, R. (2018). Greedy randomized search for scalable compilation of quantum circuits. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15* (pp. 446-461). Springer International Publishing.

[10] Rasconi, R., & Oddi, A. (2019, July). An innovative genetic algorithm for the quantum circuit compilation problem. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, No. 01, pp. 7707-7714).

[11] Li, G., Ding, Y., & Xie, Y. (2019, April). Tackling the qubit mapping problem for NISQ-era quantum devices. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 1001-1014).

[12] Tannu, S. S., & Qureshi, M. K. (2018). A case for variability-aware policies for nisq-era quantum computers. *arXiv preprint arXiv:1805.10224*.

[13] Murali, P., Baker, J. M., Javadi-Abhari, A., Chong, F. T., & Martonosi, M. (2019, April). Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems* (pp. 1015-1029).

[14] Li, S., Nguyen, K. D., Clare, Z., & Feng, Y. (2023, October). Single-Qubit Gates Matter for Optimising Quantum Circuit Depth in Qubit Mapping. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)* (pp. 1-9). IEEE.

[15] Tate, R., Farhadi, M., Herold, C., Mohler, G., & Gupta, S. (2023). Bridging classical and quantum with SDP initialized warm-starts for QAOA. *ACM Transactions on Quantum Computing*, *4*(2), 1-39.

[16] Dousti, M. J., Shafaei, A., & Pedram, M. (2014, May). Squash: a scalable quantum mapper considering ancilla sharing. In *Proceedings of the 24th edition of the great lakes symposium on VLSI* (pp. 117-122).

[17] Bahreini, T., & Mohammadzadeh, N. (2015). An MINLP model for scheduling and placement of quantum circuits with a heuristic solution approach. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, *12*(3), 1-20.

[18] Lao, L., van Wee, B., Ashraf, I., van Someren, J., Khammassi, N., Bertels, K., & Almudever, C. G. (2018). Mapping of lattice surgery-based quantum circuits on surface code architectures. *Quantum Science and Technology*, *4*(1), 015005.

[19] Lao, L., & Browne, D. E. (2022, June). 2qan: A quantum compiler for 2-local qubit hamiltonian simulation algorithms. In *Proceedings of the 49th Annual International Symposium on Computer Architecture* (pp. 351-365).

[20] Sivarajah, S., Dilkes, S., Cowtan, A., Simmons, W., Edgington, A., & Duncan, R. (2020). t|ket⟩: a retargetable compiler for NISQ devices. *Quantum Science and Technology*, *6*(1), 014003.

[21] Carrazza, S., Efthymiou, S., Lazzarin, M., & Pasquale, A. (2023, February). An open-source modular framework for quantum computing. In *Journal of Physics: Conference Series* (Vol. 2438, No. 1, p. 012148). IOP Publishing.