Effective and Efficient Parallel Qubit Mapper

Hao Fu, Mingzheng Zhu, Fangzheng Chen, Chi Zhang, Jun Wu, Wei Xie, Xiang-Yang Li, Fellow, IEEE, ACM

***Abstract*—Quantum computing has been accumulating tremendous attention in recent years. In current superconducting quantum processors, each qubit can only be connected with a limited number of neighbors. Therefore, the original quantum circuit should be converted to a hardware-dependent circuit, and this process is called qubit mapping and routing, in which typically extra SWAP gates need to be inserted. Due to a limited qubit lifetime, one of the main objectives of qubit mapping and routing is to minimize the circuit depth, which is a time-consuming process. By studying several existing greedy mappers, we extract and analyze two patterns that significantly impact the mapping and routing performance. Then, we propose a sliding window method named SWin, which dramatically reduces the computational cost with negligible performance degradation. For devices with constrained executable circuit depth, we propose SWin+, which introduces adaptive circuit slicing methods with VF2++ subgraph isomorphism initial mapping methods. Compared with the state-of-the-art greedy methods, SWin can find an effective result by up to 39% depth decrease, on average of 16% for large-scale circuits. Moreover, SWin can be easily modified to be noise-aware, while the depth reduction will yield better performance for real execution. Furthermore, SWin still performs well for various chip couplings. SWin+ significantly enhances processing efficiency, achieving improvements up to 22.3×, with an average increase of 6.1×. Concurrently, it maintains the effectiveness of the transformed circuit depth.**

***Index Terms*—Quantum computing, Mapping and routing, Parallel mapper.**

I. Introduction

Quantum computing, an emergent technology, shows considerable promise in enhancing the efficiency of critical computational tasks, such as large integer factoring [2], database searching [3], and the simulation of quantum systems [4]. Recent advancements have yielded quantum computer prototypes, including Sycamore [5], *Jiuzhang* [6], and *Zuchongzhi* [7], which have exhibited quantum supremacy by surpassing classical computational capabilities in certain tasks. Nevertheless, the progress in quantum computing is significantly hampered by current limitations in quantum experimental techniques. It is anticipated that in the near term, both the quality and quantity of qubits, as well as the fidelity of quantum operations, will remain suboptimal, characterizing what is known as the Noisy Intermediate-Scale Quantum (NISQ) era [8]. Presently, quantum devices harboring hundreds of qubits have been either constructed or emulated by entities such as IBM [9], Google [5], and Intel [10].

The process of running quantum algorithms generally involves three key steps. Initially, a quantum algorithm is transformed into a logical circuit made up of standard quantum gates [11]. This logical circuit is then adapted into a form that is compatible with the specific quantum hardware on which it will be executed. The final step involves generating a series of control signals that drive the physical quantum devices. While considerable progress has been made in understanding and optimizing the first and last steps of this process [2], [3], [4], [12], the middle step, which involves adapting the logical circuit for specific hardware, remains a significant challenge. This step is crucial for improving the performance of quantum applications, yet it still lacks robust solutions.

The present study is primarily concentrated on the intermediary step of quantum algorithm execution. The initial transformation presupposes every quantum operation can be executed among any pair of qubits, an assumption that diverges from practical constraints. Consequently, not all logical circuits are amenable to straightforward implementation on Noisy Intermediate-Scale Quantum (NISQ) devices. For instance, the logical circuit depicted in Fig.1(b) cannot be directly executed on the quantum chip illustrated in Fig.1(a), due to the lack of physical connectivity between qubits involved in each two-qubit gate. This challenge is typically surmounted through the insertion of SWAP gates, a process denoted as *the qubit mapping and routing problem* within this paper, with the algorithmic solution referred to as a *mapper*. Moreover, given the finite coherence times of qubits or device constraints on execution time, which correlates to a limited executable circuit depth, the resulting circuit must remain as shallow as possible to mitigate decoherence effects. As demonstrated in Fig.1(c) and Fig.1(d), the strategic insertion of SWAP gates significantly influences the resultant circuit depth, underscoring the critical nature of the mapping and routing strategy employed. Moreover, circuit-slicing technologies can be applied to execute deep circuits on a shallow device with a maximum allowable execution time.

Prior investigations have explored various greedy algorithms

The research is partially supported by Innovation Program for Quantum Science and Technology 2021ZD0302900, Innovation Program for Quantum Science and Technology Grant No. 2021ZD0302901, National Key R&D Program of China under Grant No. 2021ZD0110400 and China National Natural Science Foundation with No. 62132018, "Pioneer" and "Leading Goose" R&D Program of Zhejiang", 2023C01029, and 2023C01143.

Hao Fu, Mingzheng Zhu, Fangzheng Chen, Jun Wu, and Wei Xie are with the LINKE lab, University of Science and Technology of China (USTC), Hefei 230027, China (e-mail: {hflash, zmzming, fangzhengchen, jun_wu}@mail.ustc.edu.cn, xxieww@ustc.edu.cn). Chi Zhang is with School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, China and Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei 230009, China (e-mail:zhangchi@hfut.edu.cn). Xiang-Yang Li is with the LINKE lab and Hefei National Laboratory, University of Science and Technology of China (USTC), Hefei 230088, China (e-mail: xiangyangli@ustc.edu.cn). Wei Xie and Xiang-Yang Li are corresponding authors.

A preliminary version of this work titled "Effective and Efficient Qubit Mapper" was published in IEEE/ACM International Conference on Computer-Aided Design 2023, held in San Francisco, California, USA [1].

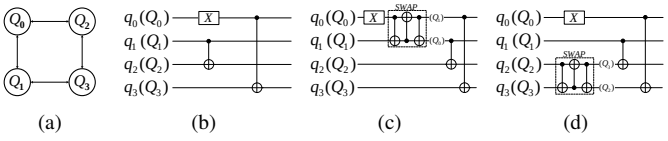


Fig. 1: Example of circuit mapping and routing. (a) Hardware coupling graph (b) The logical circuit (c) Circuit executable with circuit depth 5 (d) Circuit executable with depth 4.

aimed at minimizing the depth of the resultant circuit [13], curtailing the quantum gate count [14], [15], or mitigating the impact of quantum device noise [16]. While these greedy mappers exhibit commendable efficiency, the depth of the transformed circuit often deviates significantly from the optimal. Furthermore, several studies focused on achieving either gate count or circuit depth optimality. For instance, OLSQ [17] leverages an SMT solver [18], whereas TOQM [19] employs an A^* search algorithm to derive solutions that minimize circuit depth. Although these optimal mapping strategies yield optimal outcomes for circuits of limited scale, their computational complexity escalates exponentially with increasing circuit size. Thus, striking a balance between the efficiency and optimality of qubit mapping and routing poses a considerable challenge. Recent contributions [20], [21] have introduced circuit partitioning techniques for executing deep quantum circuits on devices with shallower capacities, circumventing the constraints posed by limited coherence times or executable circuit depth.

In this work, we discover two greedy patterns that make the usual greedy mappers perform less effectively, called *Immediate Execution* and *Online SWAP Insertion*. These patterns widely exist in general quantum circuits. This makes it hard for the greedy mapper to find optimal or near-optimal solutions efficiently. Based on the discoveries, we propose a smooth mapping algorithm **SWin** to efficiently find the mapping and routing result. Moreover, we present a new mapping and routing framework, which includes three components: quantum circuit partitioning, parallel sub-circuit smooth mapping, and parallel circuit connecting. This framework can flexibly balance the efficiency and effectiveness of qubit mapping in various quantum application scenarios. Compared with the existing known mappers, this framework provides considerable improvement in terms of mapping effectiveness and efficiency. On devices with limited execution time, we introduce **SWin+**, which contains adaptive circuit slicing technique and initial mapping methods alongside **SWin** to achieve better performance, i.e., more mapping and routing efficiency with a slight loss on device utilization.

Based on extensive evaluations, **SWin** significantly outperforms IBM qiskit [9] and TOQM (extensive greedy version) [19] on result circuit depth. **SWin** shows 2%-39% depth reduction, on average 16%, compared to the state-of-the-art algorithms on IBM novel Heavy hex architecture and grid chips, e.g., IBM Quito, Guadalupe, Prague and *zuchongzhi*. With the requirement for more effectiveness on circuit execution fidelity, noise-aware **SWin** performs well with real noise data in most cases than the original. We evaluate **SWin** on different couplings, e.g., Linear, Heavy hex and Grid, and the results also indicate the effectiveness of our algorithm. On shallow devices whose executable circuit depth is limited, **SWin+** shows great processing efficiency promotion by at most 22.3×, on avarage 6.1×, while maintaining the result circuit depth effectiveness.

The rest of this paper is organized as follows. We present motivating patterns and observations of novel mapping methods in Section II. In Section III, we provide our smooth mapping algorithm, new problems introduced by circuit partitioning and present a parallel mapping framework with adaptive circuit slicing method and initial mapping techniques. We evaluate our mapper and framework and analyze the performance in Section IV. We conclude this work with future directions in Section V.

II. Preliminaries

A. Background

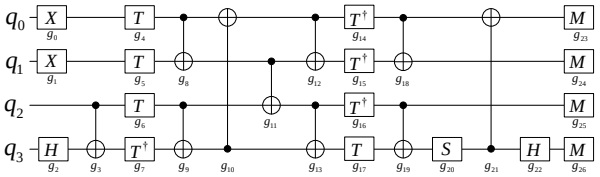
1) Qubit and Quantum Gate: In quantum computing, the fundamental element is the quantum bit, or qubit, which exists in two primary states, $|0\rangle$ and $|1\rangle$. Qubits possess the unique capability to inhabit superposition states of the form $a|0\rangle + b|1\rangle$, where $|a|^2 + |b|^2 = 1$ and $a, b \in \mathbb{C}$. Typically, qubits are initialized in the $|0\rangle$ state [22], subsequently manipulated by quantum circuits, and ultimately measured [23].

Quantum operations deploy both single and multi-qubit gates to manipulate qubit states, utilizing gates like Pauli-X, Hadamard, and CNOT. The quality of a qubit is constrained by its coherence time—the duration it maintains its quantum state before energy dissipation or environmental perturbation. Additionally, operational and measurement processes induce errors, implying that extended coherence times and reduced error rates enhance execution performance. Despite this, the inherent instability and susceptibility of qubits often derogate the performance of quantum algorithms, compounded by the limited capability of quantum devices to execute deep circuits.

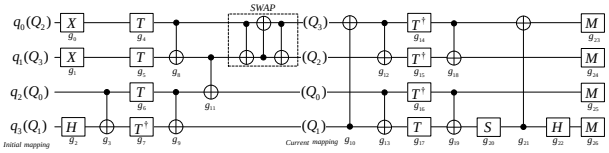
2) Quantum Circuit and Chip: Quantum algorithms are typically delineated as quantum circuits, which are temporal sequences of qubit operations. For instance, Fig. 2(a) depicts the schematic of a quantum adder's circuit, comprising a series of quantum gates such as single-qubit gates and controlled-NOT (CNOT) gates. Terminal measurement operations ($g_{23} \sim g_{26}$) are enacted to extract the computational outcomes.

In the context of Noisy Intermediate-Scale Quantum (NISQ) devices, the prevalent quantum chip architectures are illustrated in Fig. 3. For the execution of a CNOT gate, the involved qubits require a direct physical link on superconducting quantum chips. However, quantum algorithms conventionally presuppose the feasibility of operations between any qubit pair. Current quantum device technologies, regrettably, fall short of fully accommodating this ideal, owing to inherent limitations.

3) Quantum Circuit Compiling: Quantum circuits require compilation for device compatibility and performance enhancement, a process that involves several non-trivial steps. Initially, virtual circuit processing is undertaken, entailing the decomposition of complex gates and the optimization of the circuit through gate merging or elimination. Then, a mapping

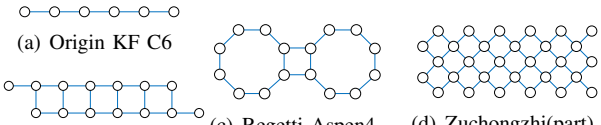


(a) The quantum adder circuit



(b) The mapping result of quantum adder circuit

Fig. 2: A typical quantum circuit and compiling result on Fig. 1(a): the horizontal line represents the qubit, and the square and the connected line with a circle across two qubits represent the quantum operation. The lines end with a measurement operation on each qubit.



(a) Origin KF C6

(b) IBM Melbourne

(c) Regetti Aspen4

(d) Zuchongzhi(part)

Fig. 3: Typical quantum chip coupling graphs: the circles indicate qubits, and the lines between the circles show that the corresponding qubits can perform two-qubit operations.

from logical to physical qubits is established to create an initial layout. However, due to chip coupling constraints, certain CNOT gates may be inexecutable in their current configuration. Thus inserting SWAP operations for logical qubit realignment is crucial, as illustrated in Fig. 2(b). This routing process iterates until the circuit is completely executable.

The final step involves additional adaptations to accommodate hardware-specific limitations, including gate set conversion and circuit depth reduction. The culmination of these procedures transforms a device-independent quantum program into an executable sequence tailored to a quantum device.

B. Qubit mapping and routing problem

We consider an input quantum circuit P with m quantum registers (represented as Q_l), and the operations in P only contain single-qubit gates and CNOT gates. We also transform this circuit P into a Directed Acyclic Graph (DAG) G_P. In G_P, each gate is represented as a node in DAG, and the dependencies between gates are represented as edges. The depth of the circuit is denoted as D_P (consider the depth of single-qubit gates and CNOT gates both to be 1 in this work), which is the depth of the longest path (critical path) in G_P. The quantum chip C contains n qubits, corresponding to a qubit set Q_p, and its *coupling graph* is denoted as a undirected graph $G_C = (Q_p, E_p)\,, E_p \subseteq Q_p \times Q_p$. The edge e_{ij} between q_i and q_j and $q_i, q_j \in Q_p$, $e_{ij} \in E_p$ exists if and only if q_i and q_j are physically connected. We have a mapping $M = f : Q_l \to Q_p$ during circuit transformation.

We denote M^I as the initial mapping, and P^T is the transformed circuit matching hardware constraints of quantum chip C. We use $\{M^I, P^T, C\}$ to denote the mapping result of P. If a circuit P can be executed on a quantum chip C starting with given initial mapping M^I by topological sorting of nodes in G_P. We call $\{M^I, P, C\}$ satisfy the *circuit executability*. Specifically, when P only consists of a single gate, we call this condition *gate executability*. As shown in Fig. 2(b), the mapping result of the quantum adder circuit satisfies the circuit executability on the chip in Fig. 1(a).

Based on the system model defined above, we then define the qubit mapping and routing problem:

Problem 1. ***Qubit Mapping and Routing problem.***
Input*: A quantum circuit P with depth D_P and quantum chip C with exact topology depicted by G_C.*
Output*: A mapping result $\{M^I, P^T, C\}$ where $\{M^I, P^T, C\}$ satisfies circuit executability and D_{P^T} is minimized.*

Problem 2. ***Min-Depth Qubit Mapping problem.***
Input*: A given quantum chip C with its qubits Q_p, quantum circuit P with its qubits Q_l and a constant k.*
Output*: A mapping $M = f : Q_l \to Q_p$ where the added depth $D_{P^T} - D_P$ is bounded by a constant k.*
Problem Hardness*: When $k = 0$, the qubit assignment problem has been proved to be NP-complete in Theorem 3.1 of [14], so this problem is NP-hard.*

Problem 3. ***Min-Depth Qubit Routing with initial mapping.***
Input*: A given mapping $M^I = f : Q_l \to Q_p$, a quantum chip C and quantum circuit P.*
Output*: A mapping result $\{M^I, P^T, C\}$ where $\{M^I, P^T, C\}$ satisfies circuit executability and D_{P^T} is minimized.*
Problem Hardness*: Parallel Token Swapping Problem (PTSP) is a special case of Min-Depth Qubit Routing. While PTSP is shown to be NP-complete [24], this problem is NP-hard.*

C. Related work

Extensive research has been conducted on qubit mapping and routing, alternatively termed quantum circuit compiling, circuit layout, circuit transformation, qubit allocation, and qubit routing [25], [26], [17], [15], [13], [27], [19], [28]. Given the NP-completeness of this challenge, various approaches have been explored, including optimization objectives, problem models, methodologies, and scalability considerations.

A significant portion of the previous works is dedicated to minimizing the number of inserted SWAPs [15], [14], [29], [25], [13], [30], [31] and augmenting SWAP parallelism, albeit without addressing the overall circuit depth. Childs et al. [27] target reducing the depth of inserted SWAPs, while Lao et al. [32] focus on the parallelism between inserted SWAPs and original gates, albeit not achieving theoretical optimality. Noise-aware approaches have also been proposed, with Tannu et al. [16] introducing a heuristic algorithm to bolster execution performance. Alternatively, works by Murali et al. [33] and Tan et al. [17] reformulate the mapping task into an SMT problem, leveraging SMT solvers to ascertain

optimal or near-optimal solutions. Furthermore, learning-based methodologies [29], [31], [34] have been employed to address the qubit mapping and routing problem.

Several studies have investigated circuit partitioning execution, where a quantum circuit is divided into subcircuits with less depth, to overcome the challenges posed by time-constrained quantum devices that limit circuit execution to a specific duration. Adrián et al. [20] devised a methodology for executing segmented circuits and reconstructing quantum states, catering to quantum devices with execution time constraints. Some research efforts [35], [36] have explored circuit simulation through partitioning based on qubits. Zulehner et al. [25] employed a strategy of dissecting circuits by layers, aiming to reduce the supplementary costs associated with quantum circuit mapping and routing. Baker et al. [21] introduced a time-sliced circuit partitioning technique, particularly for modular quantum devices. Moreover, considering the error-prone nature of qubits, some works [37], [38], [39] are focused on hardware equipped with Quantum Error Correction (QEC). These works are dedicated to the compilation of circuits encoded with QEC codes that address a variety of constraints.

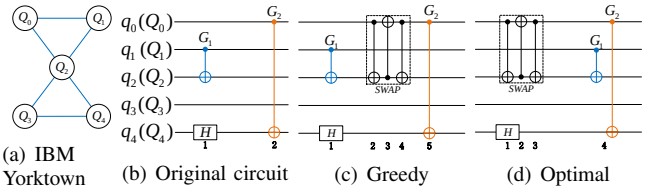
It should be noted that, although this work may be similar to that presented in TOQM [19], the primary novelty of this study lies not in the A* algorithm but rather in the sliding window approach, which is based on our discoveries regarding the two greedy patterns. The minimum-depth qubit mapping can be addressed using various methods, as it functions merely as a subroutine, including formal method-based approaches (e.g., SMT-based [17] and MAXSAT-based mapper [40]). The sliding window approach operates at a higher level. Even when our optimization goal is the number of SWAP gates or fidelity, the subroutine could involve mappers with corresponding objectives, as exemplified by the noise-aware **SWin** shown in Section IV-C2.

D. Motivation

The qubit mapper's execution procedure typically unfolds in a bifurcated manner. Initially, an initial mapping of the quantum circuit onto the quantum chip is formulated. Then, the mapper evaluates the necessity and potential locations for SWAP insertions, continuing this assessment until the circuit is rendered sequentially executable on the target quantum chip. Through an examination of various qubit mapper implementations [13], [15], [14], [17], [19], we have discerned two prevalent paradigms:

- *Immediate Execution*: If the current gate satisfies the gate executability, execute it immediately.
- *Online SWAP Insertion*: SWAP gates are not considered until no gate can be executed under the current mapping.

Greedy mapping algorithms, aimed at minimizing mapping depth, predominantly employ *Immediate Execution* and *Online SWAP Insertion* strategies to notably reduce computational time, despite maintaining polynomial complexity in circuit depth. As depicted in Fig. 4, G_1 meets the immediate executability criterion (*Immediate Execution*), whereas greedy approaches might not consider G_2 due to falling beyond the lookahead threshold (*Online SWAP Insertion*). Outcomes from such mappers often resemble the scenarios in Fig. 4(c), contrasting with the optimal mappings illustrated in Fig. 4(d).



(a) IBM Yorktown (b) Original circuit (c) Greedy (d) Optimal

Fig. 4: Examples of two patterns: when we map the circuits in Fig. 4(b) to the chip represented by Fig. 4(a), the greedy SWAP insertion strategies tend to give the answer in Fig. 4(c), Fig. 4(d) shows one of the optimal strategies.

TABLE I: The number of patterns in some typical circuits mapping and routing result on IBM Yorktown.

Circuit name	4gt11_82	4gt11_84	4gt13_92	aluv0_27	aluv1_28	aluv1_29	aluv12_33
Original depth	20	11	38	21	22	22	22
Greedy result	24	15	44	45	56	50	45
Optimal result	23	13	41	27	25	25	27
Number of patterns[1]	1	1	2	2	1	3	2

[1]: Every time the circuit depth increases due to two greedy patterns, we increase the number of patterns by 1.

An evaluation, presented in TABLE I, of a basic greedy mapper based on these principles against optimal searches (obtained by original TOQM for small-scale circuits [19]) on the IBM Yorktown chip reveals the prevalence of these patterns in greedy solutions, potentially distancing them from optimal.

III. Mapping Algorithm

In this section, we present our mapping method to solve the qubit mapping and routing problem.

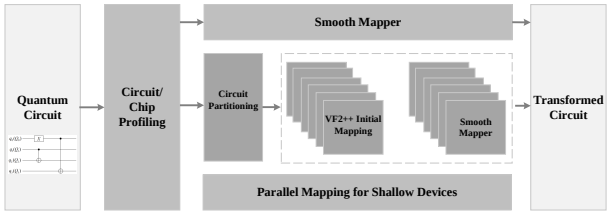


Fig. 5: Parallel mapping and routing framework.

Design Insights: We propose the smooth sliding window search algorithm, **SWin**, predicated on the principles of local search and adaptive search range. To mitigate the search time overhead in circuits with an increased number of qubits, we incorporate fine-grained parameters to optimize the trade-off between effectiveness and efficiency. To ensure the computational efficiency of **SWin**, we devise two strategies to address the exponential computational overhead: reduction to single gate and greedy execution. In light of the constrained execution time imposed on quantum devices, we introduce **SWin+** to employ an adaptive circuit partitioning technique to execute quantum circuits. This approach is complemented by integrating parallel mapping and routing with efficient initial mapping strategies to enhance mapping efficiency and execution performance. The overall mapping algorithm is shown in Fig. 5.

Algorithm 1: SWin

1 **Input** *Quantum circuit* P *and its DAG* G_P *with depth* D_p*, quantum chip* C *and searching depth* S_d;
2 **Output** *Transformed circuit* P^T*, initial mapping* M^I *and final mapping* M^F;
3 $M^I = \{Q_l^i \to Q_p^i \mid i \in \{1, 2, \cdots, |Q_l|\}\}$;
4 **if** $D_p \leq S_d$ **then**
5 P^T, M^F = *opt* (P, C, M^I);
6 **return** M^I, P^T, M^F
7 **else**
8 i = 0;
9 $M_0^I = M^I$;
10 **while** $D_p > S_d$ **do**
11 P_i = *get_front_layers* (G_P, S_d);
12 P_i^T, _ = *opt* (P_i, C, M_i^I);
13 p_i = *get_first_layer* (P_i^T);
14 M_i^F = *get_temporary_mapping* (p_i, M_i^I);
15 E_i = *get_executed_gates* (p_i, G_P);
16 G_P.remove (E_i);
17 D_p.update ();
18 $M_{i+1}^I = M_i^F$;
19 $i = i + 1$;
20 P_i = *get_front_layers* (G_P, S_d);
21 P_i^T, M^F = *opt* (P_i, C, M_i^I);
22 $p_i = P_i^T$;
23 $P^T = \{p_0, p_1, ..., p_i\}$;
24 **return** M^I, P^T, M^F

A. Sliding-Window-based Qubit Mapping and Routing

The choice of search methodology is crucial to ascertaining the optimal depth solution for qubit mapping and routing. Within each circuit layer, an exhaustive consideration of all conceivable scenarios is required, including the execution of quantum gates and the potential insertion of SWAPs for the current layer. Nonetheless, such an approach incurs exponential increases in processing time and space overhead as circuit complexity escalates, i.e., more qubits and more gates. Consequently, we introduce a sliding-window-based approach for mapping and routing, detailed in the pseudo-code presented in Algorithm 1.

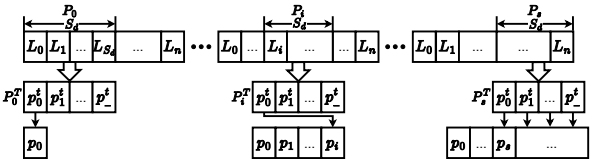
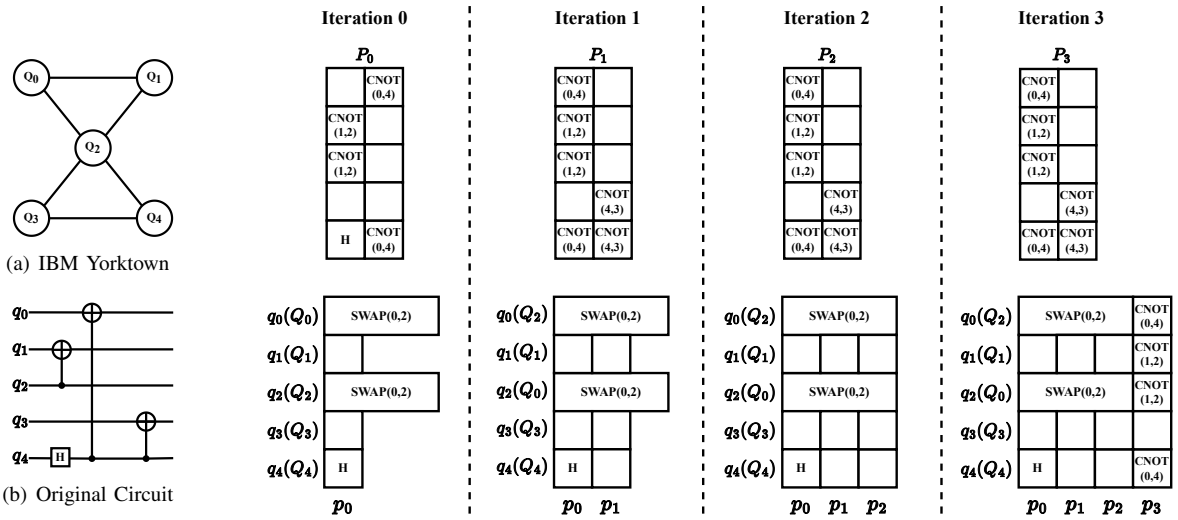


Fig. 6: **SWin** process: the sliding window moves until the depth of the remaining circuit is S_d.

Generally, with the input quantum circuit P, DAG G_P with its original depth D_P, quantum chip C and the window size parameter, denoted as S_d, our algorithm outputs the transformed circuit P^T, initial mapping M^I and final mapping M^F of the circuit on the chip C, and the $\{M^I, P^T, C\}$ satisfies the *circuit executability*. First, at the start of the circuit, we set a trivial initial mapping $M^I = \{Q_l^i \to Q_p^i \mid i \in \{1, 2, \cdots, |Q_l|\}\}$ from logical qubits to physical qubits which are located in the densest area of target chip. Then, for each S_d-depth sub-circuit P_i in the middle part of the circuit, we will find one of the optimal solutions (*opt* (P_i, C, M_i^I)), denoted P_i^T, record the first layer of the solution as p_i (*get_first_layer* (P_i^T)), and then update the circuit DAG G_P and current depth D_P. Our algorithm will iterate this process until the G_P is mapped completely. Fig. 6 provides details of this process and a straightforward example is presented in Fig. 7. A detailed explanation is listed as follows:

- Firstly we set the initial mapping $M^I = \{Q_l^i \to Q_p^i \mid i \in \{1, 2, \cdots, |Q_l|\}\}$ and compare the circuit depth D_P with the window size parameter S_d. If $D_P \leq S_d$, we obtain the optimal solution $\{M^I, P^T, C\}$ of the circuit P on chip C by *opt* (P_i, C, M_i^I)(**line 5**) and return final result.
- If $D_P > S_d$, we divide this process into following:
  - Initially, we set loop variable i = 0, and the initial mapping of first iteration $M_0^i = M^I$ (**line 8**∼**9**). As the example in Fig. 7, we map the circuit depicted in Fig. 7(b) onto the chip illustrated in Fig. 7(a), the initial mapping is set as trivial (qubit q_i in logical circuit is mapped on Q_i on the target chip). The initial size of the sliding window S_d is set to 2.
  - For i-th iteration, we take the sub-circuit P_i of the front S_d layer(s) of the circuit DAG G_P (as function *get_front_layers* (G_P, S_d) does (**line 11**). With initial mapping M_0^i, we find the optimal solution P_i^T of the circuit P_0 on chip C with subroutine *opt* (P_i, C, M_i^I) (**line 12**). We record the first layer of P_i^T as p_i (**line 13**) and get the temporary mapping M_i^F after p_i is executed **(line 14)** and take this as the initial mapping of next iteration **(line 18)**. Next, we delete the corresponding quantum gate(s) executed in p_i from G_P and update D_P (**line 15**∼**17**). We iterate this process until D_P is no longer more than S_d. As shown in Fig. 7(c), the input for the i-th iteration consists of a window of depth 2 (P_i) along with the mapping resulting from the execution of p_{i-1}. In the context of the SWAP gate execution, the last two rounds of the three rounds of iteration, referred to as iterations 1-2, are treated as having altered the mapping, although the corresponding bits are occupied. The solution for each sub-circuit is derived from the subroutine (*opt* (P_i, C, M_i^I)). For iterations 0-3, we adopt the first layer of the solution as the i-th layer of the final solution (p_i).
  - Finally, when at last $D_P = S_d$, for the remaining G_P with a circuit depth of S_d, of which the front S_d layers can also be considered as P_i (**line 20**), we still take M_i^I as the initial mapping for P_i's routing, and we obtain the optimal solution P_i^T and M_i^F (**line 21**). The result P_i^T of last iteration will be completely considered as result and M_i^F be the final mapping of the execution of the circuit (**line 22**). As shown in the fourth iteration of Fig. 7(c), because the current circuit depth aligns with



(a) IBM Yorktown

(b) Original Circuit

(c) Example mapping and routing process for circuit in (b) mapping to (a)

Fig. 7: A simple example of the **SWin** algorithm.

the sliding window size S_d, we incorporate all results from this iteration into the optimal solution (p_3).

- For circuit P and chip C, we get the mapping and routing result $P^T = \{p_0, p_1, ..., p_i\}$. Return final result M^I, P^T, M^F.

Since the initial mapping M_i^I of P_i equals the final mapping M_{i-1}^F of P_{i-1}, and G_P deletes nodes in the front layer if and only if the node has been executed in p_i, $\{M^I, P^T, C\}$ satisfies the circuit executability.

B. Key design considerations

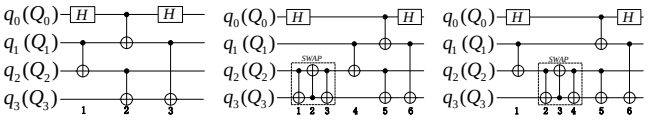
Efficiency is crucial for a qubit mapper, especially when seeking optimal or near-optimal depth solutions. Our sliding-window-based algorithm emphasizes the mapping strategy within each window. We discuss window size parameters and propose an adaptive method for adjusting window size to optimize the trade-off between result depth and time overhead. Given the potential for significant time overhead with larger circuits and quantum chips, we introduce two methods to ensure the efficiency of our mapping algorithm in worst-case scenarios. For shallow devices, we introduce adaptive circuit slicing methods and VF2++ subgraph isomorphism [41] initial mapping technique to enhance efficiency while maintaining effectiveness.

1) Min-depth circuit routing using A^*: As the subroutine of **SWin**, the selection and implementation of the *opt* function significantly influence the performance of the algorithm, particularly in terms of solving efficiency and execution effectiveness. If the implementation of this function is sub-optimal, the resulting circuit depth produced by **SWin** will be far from the optimal solution, and conversely, if the efficiency of the opt function is notably high, **SWin** will yield results with a relatively polynomial-level efficiency. As highlighted in Section II-D, significant challenges arise due to *Immediate Execution* and *Online SWAP Insertion*. One approach involves reformulating the entire circuit as an SMT problem and leveraging the z3 solver for resolution [18], yet this strategy proves highly inefficient in practice [17].

In designing an optimal circuit mapper, it is pivotal to assess the necessity of incorporating SWAP operations at each time step, alongside the immediate execution of quantum gates that meet hardware constraints. This approach entails evaluating all possible permutations of executable quantum gates and SWAPs on the chip, extending the search until the circuit's completion. Prior work [19] has demonstrated the feasibility of this approach, suggesting that heuristic functions can further improve search efficiency while maintaining optimality.

Accordingly, our proposed smooth mapper adopts the A^* search algorithm to ensure optimality, with the heuristic function following the strategy outlined in [19]. This method not only preserves optimality but also facilitates the application of various pruning techniques to enhance efficiency.

2) Smooth searching with sliding window: As depicted in Fig. 8, executing the circuit on Fig. 1(a) with SWAPs at distinct time steps maintains depth optimality, negating the need for an excessively large global search window. This revelation allows for a trade-off between efficiency and efficacy.



(a) Original circuit (b) SWAP insert strategy (c) Another strategy

Fig. 8: Sliding window size impact on optimality.

Consequently, we employ a sliding window approach, where each window resolves only the initial layer's outcome in the given sub-circuit. With window size denoted as S_d and the circuit's original depth as d, this approach effectively reduces the circuit transformation cost from $\exp(\text{poly}(d))$ to $\text{poly}(d)\exp(\text{poly}(S_d))$, given $S_d \ll d$. Employing a sufficiently large S_d ensures near-optimal results, albeit potentially

at significant time and space costs. To address this, we propose the subsequent adaptive methodologies.

3) Adaptive window size: Despite the constrained cost within its designated window, **SWin** may incur prohibitively high search costs for quantum circuits of significant scale and chips with high connectivity. To address this, we introduce T_{high} and $T_{rollback}$ to represent the node count in the search process. Exceeding T_{high} triggers a reduction in subsequent window sizes. Additionally, should search costs surpass $T_{rollback}$ during execution, a rollback mechanism is activated, terminating the current window's search and decrementing S_d by one, provided S_d is above 1. Then, we regenerate the window (sub-circuit) and re-execute our search process. Practically, for circuits with over eight logical qubits, S_d initiates at 5. A similar strategy can elevate S_d in response to T_{low}, contingent on available computational resources.

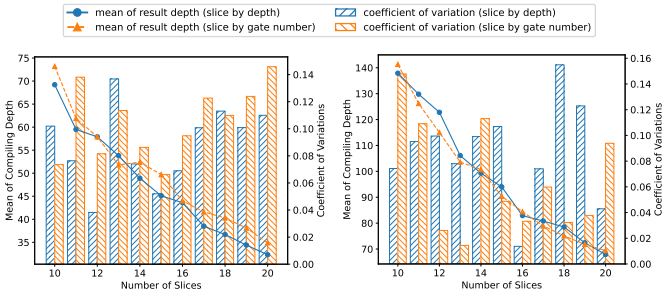
4) Efficiency-guaranteed extensions: The scalability of the A^* algorithm, particularly within the context of qubit mapping algorithms with optimality guarantees, remains a significant limitation. For example, on a 66-qubit *zuchongzhi* processor, the potential two-SWAP combinations can number in the thousands, with the total permutations being substantially higher. As previously mentioned, the time complexity of **SWin** is characterized by $\mathrm{poly}(d)\exp(\mathrm{poly}(S_d))$. Yet, the exponential factor $\exp(\mathrm{poly}(S_d))$ can become prohibitive, even for circuits with a depth of one, rendering the **SWin** algorithm vulnerable to the quantum chip's connectivity and the gate density within quantum circuits.

To address this problem, we propose two strategies: reduction to single gate and greedy execution. These are invoked when the sliding window size is reduced to one, and the computational demands remain excessive. The former strategy involves considering only a single random gate within this 1-depth window, thereby isolating chip connectivity as the sole determinant of execution time, while still accounting for circuit depth within this window. Conversely, the greedy execution approach employs a lookahead search strategy to greedily map the sub-circuit based on *Immediate Execution*, ensuring the immediate execution of any gate that meets the executability criteria. The primary additional cost associated with the greedy execution strategy is related to sub-circuit generation, requiring a practical consideration of the impact of sub-circuit depth, i.e., greedy window size S_g.

For example, let us assume it as $H(1)$, $CNOT(3,6)$, $CNOT(2,4)$, $X(5)$, and $X(8)$. Among these, the gates $CNOT(3,6)$, $H(1)$, $X(5)$, and $X(8)$ can be executed directly. The reduction to single gate strategy will randomly select one of these gates to form a new window and employ our internal search method for this 1-gate window to identify the optimal execution solution for that gate. When the selected gate is a CNOT gate that cannot be executed directly (e.g., $CNOT(2,4)$), the optimal solution corresponds to the shortest path for the related two qubits; otherwise, the gates are executed directly. In contrast, the greedy execution approach will have our algorithm directly execute $CNOT(3,6)$, $H(1)$, $X(5)$, and $X(8)$, subsequently updating the window to proceed with the algorithm.

5) Adapting shallow device: Following scenarios similar to those discussed by Perez et al. [20], quantum circuits on quantum devices can be constrained by a limited execution time, specifically a maximum allowable execution time T, which correlates to a circuit depth D_{dev}. Thus, quantum circuits are sliced into multiple sub-circuits for sequential execution. Typically, the approach involves a measure-and-reconstruct strategy where the result of the i-th sub-circuit is measured, and the quantum state is either reconstructed or loaded from QRAM before executing the (i+1)-th sub-circuit, thus avoiding qubit remapping across different sub-circuits. However, the number of sub-circuits should be as low as possible, for which many sub-circuits mean intolerant execution overhead, i.e., execution times and reconstruction or loading cost. This calls for an effective mapper for shallower devices with a limited executable circuit depth.

To address this challenge efficiently, firstly, we analyze the relationship between the original depth, the number of gates of circuits, and their mapping result depths (as shown in Fig. 9). The result indicated that (1) **SWin** shows a smooth trend of compiling result depth with circuit scaling up (either the depth or the number of gates increases); (2) The coefficient of variations differs for different circuits or scales. Based on the observations, we then introduce **SWin+**: a strategy for pre-slicing circuits and parallelizing the mapping and routing of sub-circuits. This approach not only maintains the effectiveness (i.e., the circuit depth) in the resulting circuits but also significantly enhances the efficiency of the mapping process for executing long circuits on shallow devices.



(a) cm82a_208 (b) vqe_uccsd_n6

Fig. 9: Characterization for **SWin** on two circuits: cm82a_208 and vqe_uccsd_n6. The number of slices indicates the circuit scale of sub-circuit depth or number of gates (the scale is calculated by dividing the circuit into the same number of sub-circuits). The coefficient of variation is defined as the ratio of the standard deviation to the mean value.

The initial mapping of quantum circuits may significantly impact the qubit routing process. In extreme cases, a perfect initial mapping might negate the need for any additional SWAPs. We consider further integrating initial mapping techniques to reduce the depth of the final outcomes, which could mitigate the depth increasements by circuit slicing (caused by breaking the smoothness of **SWin**). Existing initial mapping techniques typically utilize heuristic searches [15] or subgraph isomorphism approaches [26]. We adopt the VF2++ algorithm [41] to implement subgraph isomorphism for initial mappings. The algorithm's pseudo-code is shown in Algorithm 2.

Algorithm 2: SWin+: Parallel mapping and routing

Input *Quantum circuit P and its DAG G_P with depth D_p, quantum chip C and its maximum execution depth D_{dev} and searching depth S_d;*
Output *Transformed sub-circuits $P^T_{shallow} = \{\mathcal{P}^T_0, \mathcal{P}^T_1, \ldots, \mathcal{P}^T_s\}$, initial mappings for each sub-circuit $M^I_{shallow} = \{\mathcal{M}^I_0, \mathcal{M}^I_1, \ldots, \mathcal{M}^I_s\}$;*
Sample the coefficient of variation of circuit depth and number of gates $c_v(depth)$ and $c_v(gate\_num)$;
Assess an initial slice value v_{ini};
Calculate the final slice value v_{final};
Cut P to sub-circuits $P_s = \{\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_s\}$;
for *sub-circuit* $\mathcal{P}_i \in P_s$ **do**
  $\mathcal{P}_{tmp} = \emptyset$;
  while $G_{\mathcal{P}_i} \neq \emptyset$ **do**
    Add the first gate $g \in G_{\mathcal{P}_i}$ to $\mathcal{P}_{tmp}$ in topological sort of $G_{\mathcal{P}_i}$;
    Remove g from $G_{\mathcal{P}_i}$;
    if *VF($\mathcal{P}_{tmp}, C$)* **then**
      $\mathcal{M}^I_i$ = VF($\mathcal{P}_{tmp}, C$);
    else
      break;
  Use **SWin** to obtain the mapping and routing result $\mathcal{P}^T_i$ of sub-circuit $\mathcal{P}_i$ with initial mapping $\mathcal{M}^I_i$;
return $M^I_{shallow}, P^T_{shallow}, C$

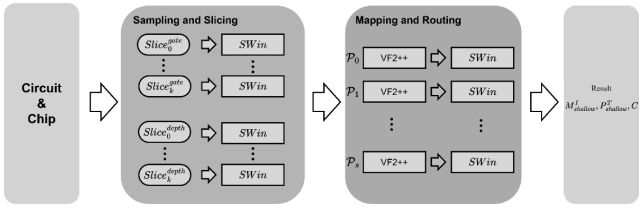


Fig. 10: **SWin+** process: with sampling result for cutting parameter, input circuits will be sliced and parallelly mapping and routing with **SWin**.

Compared to the **SWin**, the **SWin+** additionally requires the maximum depth D_{dev} supported by the device. As shown in Fig. 10, we use initial sampling values v_{ini} to assess the correlation between the mapping results and the original circuit depth/gate count, with sampling sub-circuits parallelly. Then, we determine the final circuit slicing parameters v_{final} and then cut the circuit for further processing using the VF2++ algorithm [41] alongside **SWin**. Finally, we merge the results and return $M^I_{shallow}, P^T_{shallow}, C$.

Sampling and Slicing Step: Initially, we estimate a sampling initial value $v_{ini} = \{v_d, v_g\}$, where v_d and v_g represent the circuit slicing depth and gate count, respectively, aiming to approximate the resulting circuit's depth to the device's maximum executing time D_{dev}. The relationship is given by:

$$v_d, v_g = \frac{D_{dev}}{|E|/|Q_p|},$$

where $|E|$ and $|Q_p|$ denote the number of edges and qubits in the device's connectivity graph, respectively. By sampling a subset of sub-circuits with depths v_d and gate counts v_g with **SWin**, we obtain the depth D_{sample} and coefficients of variations $c_v(depth)$ and $c_v(gate\_num)$, which is defined as the ratio of the standard deviation to the mean value. Given that results from **SWin** suggest a linear relationship between the original circuit depth and gate count versus the resulting circuit depth, we set the sampling result estimation distance $dist$ as:

$$dist = \alpha \times v_{ini} \times \left| \frac{D_{dev}}{D_{sample}} - 1 \right|,$$

where $\alpha = 1$ for slicing with depth and 2 for the number of gates. The sampling range for v_{final} is set between:

$$[v_{ini} - (1-\sigma) \times dist, v_{ini} + (1+\epsilon) \times dist].$$

In our evaluations, σ and ϵ are typically set as 0.5 and 1, respectively. The v_{final} in the sampling range that brings the average depth closest to D_{dev} without exceeding it is chosen from the sampling results. The circuit is then divided into sub-circuits based on the chosen v_{final}.

Mapping and Routing Step: For each sub-circuit $\mathcal{P}_i$, the initial mapping is determined using the VF2++ algorithm [41] applied to a temporary circuit $\mathcal{P}_{tmp}$. As gates are added to $\mathcal{P}_{tmp}$ following the topological order within $\mathcal{P}_i$, we check for subgraph isomorphism with the device connectivity graph C until $\mathcal{P}_{tmp}$ and C no longer match (VF($\mathcal{P}_{tmp}, C$) = False). The last successful mapping result $\mathcal{M}^I_i$ from VF($\mathcal{P}_{tmp}, C$) is then used as the initial mapping to solve the final sub-circuit mapping result $\mathcal{P}^T_i$. Finally, the algorithm returns outputs $M^I_{shallow}, P^T_{shallow}, C$, where $P^T_{shallow} = \{\mathcal{P}^T_0, \mathcal{P}^T_1, \ldots, \mathcal{P}^T_s\}$ and $M^I_{shallow} = \{\mathcal{M}^I_0, \mathcal{M}^I_1, \ldots, \mathcal{M}^I_s\}$.

IV. Evaluation

We assess the performance of **SWin** using benchmark circuits from QASMBench [42], a widely recognized benchmark suite for the NISQ era. We compare **SWin** against several leading methods, including IBM Qiskit [9] and TOQM [19]. Detailed experimental results are presented in Sections IV-B and IV-C. Our key findings are summarized as follows:

- **SWin** efficiently obtains near-optimal solutions for small-scale circuits with a sliding window size of 8.
- **SWin** consistently outperforms other state-of-the-art greedy algorithms for large-scale circuits, achieving up to a 39% reduction in the depth of transformed circuits, with an average reduction of 16% compared to TOQM.
- **SWin** is adaptable as a noise-aware mapping algorithm. It significantly enhances overall circuit execution fidelity across most benchmarks, with the depth reductions achieved also contributing to substantial fidelity improvements during actual quantum circuit executions.
- Despite varying chip connectivities, **SWin** significantly reduces the depth of result circuits, particularly excelling on the IBM Heavy Hex architecture.
- **SWin+** can greatly reduce the time overhead of the mapping and routing process on shallow devices by at most 22.3× while maintaining the result circuit depth effectiveness.

TABLE II: Comparision of total qubit mapping methods.

Benchmark name	n	d	g	IBM [1]			TOQM			**SWin** (reduction to single gate)			**SWin** (greedy execution) [2]		
				Depth	CNOT	RT	Depth	CNOT	RT	Depth	CNOT	RT	Depth	CNOT	RT
qaoa_n3	3	10	14	22	6	0.005	**16**	6	**0.003**	**16**	6	0.016	-	-	
fredkin_n3	3	11	18	34	6	0.007	18	6	**0.003**	**17**	6	0.022	-	-	-
wstate_n3	3	17	24	30	6	0.005	24	9	**0.004**	**23**	6	0.021	-	-	-
cat_state_n4	4	4	4	9	3	0.005	**7**	3	**0.003**	**7**	3	0.004	-	-	-
hs4_n4	4	5	12	13	3	0.005	**11**	3	**0.003**	**11**	3	0.013	-	-	-
bell_n4	4	9	19	27	6	0.005	19	6	**0.004**	**16**	12	0.028	-	-	-
adder_n4	4	11	21	45	9	0.006	24	9	**0.004**	**21**	18	0.055	-	-	-
variational_n4	4	25	40	54	3	0.009	36	3	**0.004**	**33**	12	0.053	-	-	-
basis_test_n4	4	53	85	155	18	**0.012**	84	9	0.044	**61**	12	0.068	-	-	-
vqe_uccsd_n4	4	128	154	370	87	**0.026**	198	69	0.121	**135**	12	0.069	-	-	-
basis_trotter_n4	4	733	1264	1901	150	**0.105**	1144	105	0.994	**741**	12	0.465	-	-	-
qec_en_n5	5	14	22	39	9	0.007	23	9	**0.003**	**19**	9	0.012	-	-	
error_correctiond3	5	77	113	191	27	**0.016**	93	18	0.020	**83**	18	0.058	-	-	-
qaoa_n6	6	65	146	101	45	**0.01**	**97**	16	0.42	99	72	3.64	99	72	2.29
vqe_uccsd_n6	6	1325	1505	2193	1017	**0.11**	2139	479	11.28	**1405**	315	10.52	1654	564	18.00
dnn_n8	8	97	520	209	66	**0.03**	208	35	1.36	**169**	210	27.38	169	210	17.52
cm82a_208	8	337	650	1001	687	**0.05**	828	203	5.35	**726**	1062	24.10	726	1062	24.11
rd53_251	8	712	1291	2057	1380	**0.10**	1704	374	10.73	**1512**	1980	41.58	1512	1980	41.57
vqe_uccsd_n8	8	6451	7175	10792	6075	**0.75**	9993	1804	54.16	**7618**	2451	157.06	7820	2925	114.58
ising_n10	10	41	235	126	102	**0.02**	84	37	0.50	**80**	165	26.00	80	165	26.37
adder_n10	10	95	134	178	123	**0.02**	159	56	0.75	**147**	210	5.75	156	201	6.33
sat_n11	11	387	597	846	630	**0.05**	691	167	4.05	634	966	26.43	**616**	843	17.70
z4_268	11	1644	3073	5104	3903	**0.36**	4129	1115	27.61	**3615**	5166	153.60	3615	5166	154.21
cycle10_2_110	12	3386	6050	10416	8403	**0.66**	8402	2163	53.89	**7235**	10674	281.37	7235	10674	276.33
adr4_197	13	1839	3439	5788	4740	**0.30**	4754	1364	30.89	**3963**	5493	143.01	4019	5802	161.87
multiplier_n15	15	228	492	560	612	**0.04**	491	197	2.98	**457**	831	41.03	461	831	39.87
wstate_n27	27	55	105	399	284	**0.05**	173	110	0.99	**116**	225	4.34	116	225	4.74
vqe_real_amplitudes_30	30	31	89	238	183	**0.04**	162	68	1.04	**99**	228	3.66	99	228	4.10
vqe_real_amplitudes_60	60	61	179	663	522	**0.07**	310	438	14.11	**235**	585	160.81	235	585	164.40

[1] The results of IBM qiskit [9] are generated by applying all transpiling methods and choosing the result with the lowest depth. [2] For small circuits, **SWin** with reduction to single gate and greedy execution strategies show the same performance, due to the window size never decreasing to 1. n, d, g: number of qubits, circuit depth and number of gates. Depth, CNOT, RT: result circuit depth, number of additional CNOT gates and runtime (in seconds).

A. Methodology

Dataset: We utilize quantum circuits from QASMBench [42] and Variational Quantum Eigensolver (VQE) real amplitude circuits produced using IBM Qiskit [9] for our evaluations. Additionally, for the analysis of gate density impact, we generate random circuits with a fixed gate density using the QUEKO benchmark suite [43].

Hardware Model: We conduct an assessment of our mapping and routing algorithm across various quantum devices to demonstrate its adaptability and efficiency. For circuits comprising fewer than five qubits, we utilize IBM Quito. Circuits ranging from six to sixteen qubits are evaluated on IBM Guadalupe, while those containing seventeen to thirty-three qubits are tested using IBM Prague. Additionally, we explore the performance of our algorithm on a 60-qubit Variational Quantum Eigensolver (VQE) circuit implemented on *Zuchongzhi*. To analyze the impact of chip coupling, we employ coupling graphs typical of linear layouts, IBM's Heavy Hex, and grid couplings.For shallow devices, we still use 16-qubit linear, heavy hex and grid couplings, with the limited device executable depth $D_{dev} = 100$.

Evaluation Platform: Our experiments are performed on Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz, with 128GB DDR4 memory. The operating system is Ubuntu 20.04.

Algorithm Configuration: For small-scale circuits, we fix the sliding window size to 8. Based on our sensitivity analysis in Section IV-C, we empirically set the initial window size as 5, s_{dmax} and s_g as 10 and adopt the adaptive method with $T_{rollback} = 1700$, $T_{high} = 1000$ and $T_{low} = 500$. For **SWin+**, we generate several v_final values for parallel experiments and take the one with the lowest depth as the final result.

Baselines: We compare our work with two greedy mapper solutions. The two greedy mappers are IBM qiskit [9] (denoted as IBM, the results are generated by applying all transpiling methods and choosing the lowest depth) and extended greedy mapper from [19] (denoted as TOQM). These three mappers are initialized with a trivial initial mapping. For shallow devices, we first compile the circuits by **SWin** with VF2++ method [41] for initial mapping and then perform the slicing process, using these methods as the baseline.

Metrics: In our study, we systematically compare the depth of the resultant circuits and processing times across a range of scenarios. Additionally, within the context of noise-aware mapping experiments, we assess the fidelity of the results utilizing noise data from IBM Guadalupe to further validate our algorithm's robustness and practical applicability under realistic quantum computing conditions. For shallow devices, besides result depth and processing time, we introduce an additional metric, Device Utilization Ratio (DUR), which denotes the overall device execution time utilization and can be calculated by:

$$\text{DUR of } \textbf{SWin+} = \frac{\text{Depth } (\textbf{SWin+})}{D_{dev} * S_{num}},$$

where S_{num} denotes the number of sub-circuits.

B. Experimental Results

Overall Result. With a constrained sliding window size $S_d = 8$, **SWin** can consistently find near-optimal solutions. Despite the limitations imposed by the window size, **SWin** outperforms other state-of-the-art greedy algorithms in terms of performance for most large-scale circuits, as detailed in Table

TABLE III: Results on shallow devices.

Benchmark name	Coupling	SWin+				SWin			
		S_{num} [1]	RT	Depth	DUR(%)	S_{num}	RT	Depth	DUR(%)
vqe_uccsd_n6	Linear	15	3.41	1345	89.67	14	**2.71**	**1341**	95.79
	Heavy hex	17	**7.38**	1378	81.06	14	10.62	**1352**	96.57
	Grid	14	**12.86**	1328	94.86	14	116.11	**1327**	94.79
cm82a_208	Linear	11	**8.14**	781	71.00	8	17.04	**755**	94.38
	Heavy hex	8	**9.40**	**728**	91.00	8	27.16	732	91.50
	Grid	7	**72.10**	601	85.86	6	104.38	**600**	100.00
vqe_uccsd_n8	Linear	117	**6.69**	6827	58.35	66	55.56	**6547**	99.20
	Heavy hex	97	**13.05**	6722	69.30	66	146.01	**6562**	99.42
	Grid	68	**133.89**	6477	95.25	65	1038.66	**6469**	99.52
sat_n11	Linear	9	**8.93**	699	77.67	7	21.39	**695**	99.29
	Heavy hex	7	**16.70**	**591**	84.43	7	32.66	631	90.14
	Grid	5	181.88	463	92.80	5	**172.22**	**462**	92.40
z4_268	Linear	48	**12.62**	3884	80.92	37	97.44	**3693**	99.81
	Heavy hex	40	**19.06**	**3527**	88.17	36	146.04	3584	99.56
	Grid	34	**234.02**	**2942**	86.53	30	754.49	2966	98.87
cycle10_2_110	Linear	113	**9.13**	8097	71.65	76	203.35	**7517**	98.91
	Heavy hex	79	**15.36**	**6947**	87.94	72	281.11	7132	99.06
	Grid	67	**184.94**	**5946**	88.75	60	1309.92	5965	99.42
adr4_197	Linear	56	**10.84**	4418	78.89	42	112.61	**4146**	98.71
	Heavy hex	44	**14.22**	**3848**	87.45	40	143.81	3975	99.38
	Grid	37	**161.50**	**3285**	88.78	34	806.24	3338	98.18
multiplier_n15	Linear	7	**17.90**	**504**	72.00	7	27.10	605	86.43
	Heavy hex	5	**26.72**	**420**	84.00	5	33.29	421	84.20
	Grid	4	**443.11**	**336**	85.50	4	443.88	337	84.25

[1] The number of sub-circuits. RT, Depth, DUR: processing time (in seconds), result circuit depth and overall device execution time utilization. For **SWin+**, we only focus on circuit depth of more than 300. RT of **SWin+** indicates the maximum processing time of the sum of sampling, each sub-circuit mapping and routing time.

II. Specifically, **SWin** can reduce the depth of the transformed circuits by up to 39%, with an average reduction of 16% compared to TOQM. This reduction significantly enhances the quantum execution time, i.e., the circuit depth, where **SWin** incurs only a constant factor increase in computational time relative to traditional greedy methods.

For large-scale circuits and target chips, the flexibility of **SWin** allows for adjustments using reduction to single gate and greedy execution strategies to achieve more efficient and effective outcomes. Additionally, the noise-aware version of **SWin** demonstrates substantial improvements in fidelity across most circuits. Even as chip connectivity varies, **SWin** continues to outperform baseline algorithms by as much as 39% on the IBM Heavy Hex architecture, particularly on chips characterized by lower connectivity.

For shallow devices, as shown in Table. III, **SWin+** outperforms baseline methods significantly in terms of processing time, shortening it by 22.3× at most, on average 6.1×, with a slight loss on device utilization ratio in most cases, on average 83.0%. For result circuit depth, **SWin+** also performs well, maintaining the effectiveness of **SWin**, i.e., the result circuit depth decreased at most by 16.7% while at worst increased by 7.8%. For some circuits with a shallow initial depth (such as sat_n11, multiplier_n15), the time consumption of **SWin+** increased. This is mainly because of the two time-consuming **SWin** executing steps in **SWin+** process.

C. Sensitivity Analysis

1) Compared with formal methods: We compare the results of **SWin** executed on small-scale quantum circuits (utilizing a trivial initial mapping) with OLSQ [17] which is a formal method-based mapper, as illustrated in Table IV. The results demonstrate that, although the sliding window mapping algorithm employed by **SWin** remains fundamentally a greedy strategy, it can achieve commendable results on small-scale circuits, with some instances yielding faster optimal solutions. It is because that for certain circuits, the local optimal solutions during the circuit mapping process align with the global optimal solutions, and the trivial initial mapping is also one of the initial mappings for these optimal solutions. Furthermore, our approach exhibits a substantial improvement in efficiency compared to OLSQ, particularly as the number of qubits and circuits increases.

TABLE IV: Comparison with formal method-based mapper.

Benchmark name	n	d	OLSQ		**SWin**	
			Depth	RT	Depth	RT
qaoa_n3	3	10	10	1.69	16	0.02
fredkin_n3	3	11	11	2.43	17	0.02
wstate_n3	3	17	17	5.77	23	0.02
cat_state_n4	4	4	4	0.26	7	0.01
hs4_n4	4	5	5	0.57	11	0.01
bell_n4	4	9	9	1.90	16	0.03
adder_n4	4	11	11	16.07	21	0.06
variational_n4	4	25	25	16.51	33	0.05
basis_test_n4	4	53	53	146.08	61	0.07
vqe_uccsd_n4	4	128	128	1023.06	135	0.07
basis_trotter_n4	4	733	N/A[1]	N/A [1]	741	0.47
qec_en_n5	5	14	14	3.65	19	0.01
error_correctiond3_n5	5	77	77	943.60	83	0.06

[1] The circuit basis_trotter_n4 costs more than three days without returning a result.

2) Noise-aware mapping: The A^* heuristic function of **SWin** is readily adaptable to incorporate noise-aware capabilities. In scenarios where the depth implications of two SWAP insertion strategies are equivalent, the estimated fidelity serves as a decisive tie-breaker. Utilizing noise data from IBM

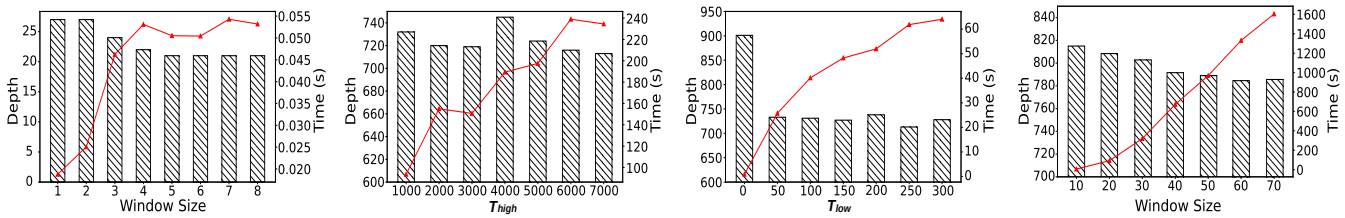
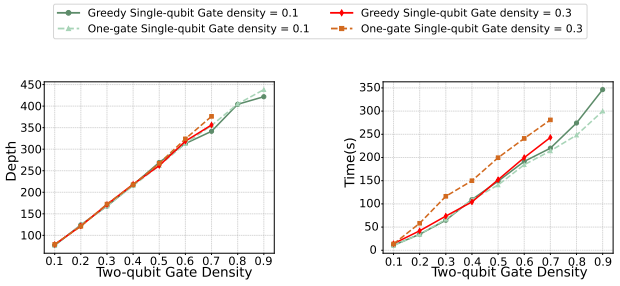


Fig. 11: Impact of sliding window size on the compiling result of circuits and T_{high} and T_{low} on large-scale circuits. (a) Impact of window size on result depth and processing time (b) Impact of T_{high} (c) Impact of T_{low} (d) Impact of greedy window size.

Guadalupe, we conduct further experiments on benchmark circuits to compare the performance of the original **SWin** with its noise-aware variant, as detailed in Table V. The findings reveal that the overall fidelity is effectively enhanced across most benchmarks. This reduction in circuit depth also contributes to significant fidelity improvements during actual execution of quantum circuits.

TABLE V: Noise-aware results.

Benchmark name	Original **SWin**		Noise-aware **SWin**	
	Depth	Fidelity	Depth	Fidelity
adder_n10	147	0.019	**138**	**0.083**
qaoa_n6	99	**0.266**	**94**	0.258
ising_n10	**80**	**0.057**	113	0.028
vqe_uccsd_n6	**1405**	**1.814e-07**	1584	6.284e-08
cm82a_208	726	3.187e-07	**721**	**8.976e-06**
sat_n11	616	7.529e-07	**593**	**1.878e-05**
multiplier_n15	467	1.215e-06	**427**	**5.252e-05**
dnn_n8	**169**	0.007	170	**0.016**
rd53_251	1512	1.221e-16	**1484**	**8.526e-10**
z4_268	3615	2.568e-33	**3471**	**1.666e-25**
adr4_197	3963	2.456e-35	**3910**	**6.888e-29**
vqe_uccsd_n8	**7618**	**1.030e-34**	7930	7.475e-40



(a) Impact on depth (b) Impact on time

Fig. 12: Impact of gate density on reduction to single gate and greedy execution strategies' result depth and processing time.

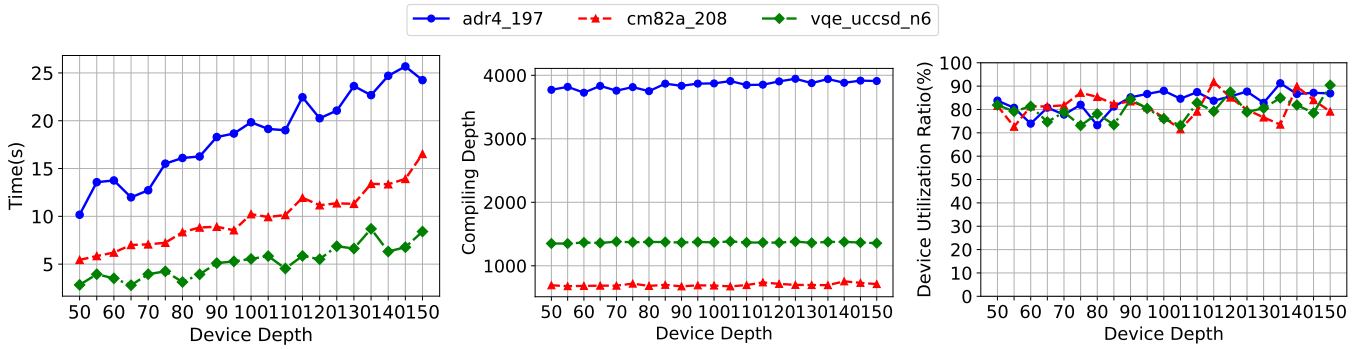
3) Comparison of reduction to single gate and greedy execution strategies: To examine the impact of two scalable strategies, we varied the gate density from 0.1 to 1 for both single-qubit and two-qubit gates, with results depicted in Fig. 12. The performance metrics used to evaluate these strategies include time overhead and the depth of the resultant circuit. The findings demonstrate **SWin**'s efficiency, characterized by a linear increase in performance as gate density escalates. Similarly, the performance related to circuit depth suggests that the strategy of considering only one gate at a time resembles a greedy approach more than a search method. Circuits with fixed densities were generated using QUEKO [43], each with a depth of thirty. For each specified density of single-qubit and two-qubit gates, ten random circuits were produced and mapped to IBM Guadalupe, with the average depth and processing time serving as metrics for effectiveness and efficiency. As the density of two-qubit gates increased, the greedy execution strategy demonstrated greater efficiency, maintaining similar performance in terms of circuit depth. The Immediate Execution scheduling strategy facilitates the more efficient performance of greedy execution, particularly at higher single-qubit gate densities, because it allows for the immediate execution of all single-qubit gates without the need to consider the current mapping status.

4) Chip connectivity: The influence of chip connectivity on circuit performance is presented in Table VI. The results demonstrate that **SWin** effectively reduces circuit depth across various chip couplings. To quantify the performance of large-scale circuit mapping, we employ the Depth Reduction Ratio (DRR). This metric calculates the efficiency of **SWin** in decreasing circuit depth and is defined as follows:

$$\text{DRR of } \textbf{SWin} = \frac{\text{Depth (TOQM)} - \text{Depth (}\textbf{SWin}\text{)}}{\text{Depth (TOQM)}},$$

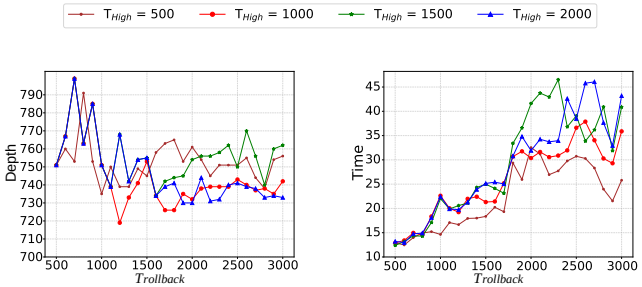
where Depth (TOQM) means the result obtained by TOQM, a higher depth reduction ratio indicates better performance. Experiments are conducted on three characteristic chip couplings: Linear, Heavy Hex, and Grid. These configurations represent varying levels of connectivity, corresponding to technologies from IBM, Google [5], and USTC [7] in superconducting quantum computers, respectively. Notably, **SWin** tends to be more effective on chips with lower connectivity as compared to TOQM. This increased effectiveness can be attributed to the relatively constrained search space available on chips with lower connectivity, which allows our algorithm to utilize a medium-sized sliding window and achieve results with reduced circuit depth. Specifically, for circuits such as wstate_n27 and vqe_real_amplitudes, traditional greedy strategies often struggle to identify an optimal initial mapping, whereas **SWin** consistently outperforms these approaches.

5) Window size impact: Fig. 11(a) illustrates the impact of the sliding window size S_d used in **SWin** on the performance of small-scale quantum circuits. The results indicate that as S_d increases, the depth of the resulting circuit decreases while the linear processing time correspondingly increases. With an



(a) Impact on processing time (b) Impact on result depth (c) Impact on DUR

Fig. 13: Impact of D_{dev} on result depth, processing time and DUR on three circuits: adr4_197, cm82a_208 and vqe_uccsd_n6. We change D_{dev} from 50 to 150 for IBM Guadalupe chip architecture, with VF2++ [41] initial mapping method.



(a) Impact on depth (b) Impact on time

Fig. 14: Impact of $T_{rollback}$ and T_{high} on result circuit depth and processing time: with a fixed T_{high}, we set $T_{rollback}$ from 500 to 3000.

TABLE VI: Results on different chip couplings.

Benchmark name	Coupling	TOQM		**SWin**		DRR
		Depth	Time	Depth	Time	
cm82a_208	Linear	867	6.24	747	14.28	0.14
	Heavy hex	828	5.59	726	23.96	0.12
	Grid	649	4.87	616	137.94	0.05
sat_n11	Linear	704	3.93	645	17.92	0.08
	Heavy hex	691	4.05	634	25.45	0.08
	Grid	532	2.64	483	199.07	0.09
wstate_n27	Linear	54	0.003	54	0.33	0.00 [2]
	Heavy hex	173	0.98	116	4.25	0.33
	Grid	98	2.08	88	64.96	0.10
vqe_real_amplitudes [1]	Linear	31	0.004	31	0.39	0.00 [2]
	Heavy hex	162	1.05	99	3.57	0.39
	Grid	79	0.56	69	31.11	0.13

[1] The circuit vqe_real_amplitudes means a 30-qubit and 31-depth VQE circuit generated by IBM qiskit [9]. [2] These two circuits can be mapped optimal on a target chip with linear couplings with a trivial initial mapping.

appropriately selected window size, **SWin** demonstrates both effectiveness and efficiency. For scenarios requiring optimal mapping results, the sliding window size S_d is set equal to the depth of the circuit to ensure the best possible outcomes.

For large-scale circuits, it was observed that S_d did not provide sufficient smoothness for time processing; therefore, we evaluated **SWin** using T_{high} and T_{low} as parameters. Initially, T_{high} was varied from 1000 to 7000 in increments of 1000, with T_{low} set at 500. Additionally, T_{low} was adjusted from 0 to 300 while maintaining $T_{high} = 2000$ during the processing of the cm82a_208 circuit. Then, we examined changes in the depth of the results and the processing time as T_{high} and T_{low} increased to assess the scalability of **SWin**. From Fig. 11(b) and 11(c), it is evident that the result circuit depth generally decreases while processing time exhibits a linear increase as T_{high} and T_{low} are raised.

Furthermore, the window size for the greedy execution strategy must be carefully considered in our evaluation. To isolate the effects of other parameters, such as T_{high}, T_{low}, and $T_{rollback}$, we propose substituting the search algorithm in each sliding window with this greedy strategy. As illustrated in Fig. 11(d), an increase in window size leads to a decrease in the result depth of the greedy method, which then plateaus, while processing time exhibits a linear increase. The performance in terms of effectiveness highlights the potential of greedy methods, particularly as the look-ahead strategy mirrors our sliding window approach when applied at a medium window size scale. The increase in processing time is attributed to switching windows cost: sub-circuits are iteratively generated with lengths corresponding to the current window size.

Finally, we show the result of the impact of $T_{rollback}$ and T_{high}. Fig. 14 shows the overall depth decreasing and time increasing trend as $T_{rollback}$ increases. As aforementioned in Section. III-B3, when the cost inside a window is more than T_{high}, we will reduce the window size of *next* window by one. When the cost is more than $T_{rollback}$, we immediately stop the mapping process in this window and reduce the window size of *current* window by one, then re-execute this window. Intuitively, $T_{rollback}$ should be set exceed T_{high}, otherwise T_{high} will be invalidated. We set T_{high} as 500, 1000, 1500, 2000 and $T_{rollback}$ from 500 to 3000 with step 100. The fluctuation of the curves in Fig. 14 reflects the different sensitivity of fixed values on window size, impacting the result circuit depth and processing time.

6) Limited device executable depth for shallow device: To investigate the impact of limited device executable depth D_{dev}, we change D_{dev} from 50 to 150 with IBM Guadalupe chip architecture (16-qubit heavy hex architecture), with VF2++ [41] initial mapping method. We test three circuits: adr4_197, cm82a_208 and vqe_uccsd_n6. As shown in Fig. 13, the compiling time increases linearly with D_{dev} increasing, and the result depths have little changes. The overall Device

Utilization Ratio (DUR) maintains at least 70%, averagely around 80%.

In Fig. 13(a), with more device executable depth D_{dev}, the scale of the sub-circuit will be larger (whether the depth is deeper or the number of gates is greater), which naturally leads to an increase in compiling time. Due to the effectiveness of the initial mapping method, when the device executable depth is small, the result depth after compilation decreases, as shown in Fig. 13(b). In Fig. 13(c), the curves of the relationship of DUR and D_{dev} show fluctuations, for the randomness of sampling may significantly influence the slicing parameter and further impact the DUR of results. Each sub-circuit's structure and the initial mapping using the VF2++ method are also important factors influencing the results.

V. Conclusion and discussion

In this study, we introduce a qubit mapping and routing algorithm to optimize the balance between effectiveness and efficiency in quantum circuit compilation. We start by examining the gaps between the results from conventional greedy mappers and the optimal solutions, identifying inefficiencies. These insights lead to the discovery of two patterns that significantly prevent achieving optimal results. Based on these findings, we develop **SWin**, an advanced algorithm designed to efficiently achieve near-optimal mapping outcomes through two efficiency-guaranteed strategies: reduction to single gate and greedy execution. Empirical evaluations show that **SWin** significantly improves effectiveness, achieving a 2%-39% reduction in circuit depth compared to leading greedy methods. To address the impact of chip noise, **SWin** can adapt into a noise-aware variant, with only a slight reduction in effectiveness. Even with variations in target chip coupling, **SWin** consistently achieves significant reductions in result circuit depth. Especially in shallow devices with limited executable circuit depth, the enhanced version **SWin+** significantly improves the efficiency of the mapping and routing processes while maintaining the quality of results in terms of circuit depth.

Limitations and Future Work: The main challenge is balancing effectiveness and efficiency in qubit mapping and routing. The optimal solution method is still too costly and time-consuming, limiting **SWin**'s ability to achieve both goals in some scenarios. Additionally, the performance of our efficiency-guaranteed strategies, reduction to single gate and greedy execution, may be affected on highly connected chips. Future efforts will focus on refining these strategies to overcome connectivity-related challenges and enhance the overall effectiveness of the algorithm.

References

[1] H. Fu, M. Zhu, J. Wu, W. Xie, Z. Su, and X.-Y. Li, "Effective and efficient qubit mapper," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[3] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[4] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, pp. 1–7, 2014.

[5] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[6] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu *et al.*, "Quantum computational advantage using photons," *Science*, vol. 370, no. 6523, pp. 1460–1463, 2020.

[7] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan *et al.*, "Strong quantum computational advantage using a superconducting quantum processor," *Physical review letters*, vol. 127, no. 18, p. 180501, 2021.

[8] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[9] M. S. ANIS, Abby-Mitchell, H. Abraham, AduOffei, and G. A. e. a. Rochisha Agarwal, "Qiskit: An open-source framework for quantum computing," 2021.

[10] G. G. Guerreschi, J. Hogaboam, F. Baruffa, and N. P. Sawaya, "Intel quantum simulator: A cloud-ready high-performance simulator of quantum circuits," *Quantum Science and Technology*, vol. 5, no. 3, p. 034007, 2020.

[11] T. Sleator and H. Weinfurter, "Realizable universal quantum logic gates," *Physical Review Letters*, vol. 74, no. 20, p. 4087, 1995.

[12] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, "Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms," *Journal of magnetic resonance*, vol. 172, no. 2, pp. 296–305, 2005.

[13] A. Zulehner, S. Gasser, and R. Wille, "Exact global reordering for nearest neighbor quantum circuits using a*," in *International conference on reversible computation*. Springer, 2017, pp. 185–201.

[14] M. Y. Siraichi, V. F. d. Santos, S. Collange, and F. M. Q. Pereira, "Qubit allocation," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, 2018, pp. 113–125.

[15] G. Li, Y. Ding, and Y. Xie, "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*, 2019, pp. 1001–1014.

[16] S. S. Tannu and M. K. Qureshi, "Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 987–999.

[17] B. Tan and J. Cong, "Optimal layout synthesis for quantum computing," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[18] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[19] C. Zhang, A. B. Hayes, L. Qiu, Y. Jin, Y. Chen, and E. Z. Zhang, "Time-optimal qubit mapping," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 360–374.

[20] A. Pérez-Salinas, R. Draškić, J. Tura, and V. Dunjko, "Shallow quantum circuits for deeper problems," *Physical Review A*, vol. 108, no. 6, p. 062423, 2023.

[21] J. M. Baker, C. Duckering, A. Hoover, and F. T. Chong, "Time-sliced quantum circuit partitioning for modular architectures," in *Proceedings of the 17th ACM International Conference on Computing Frontiers*, 2020, pp. 98–107.

[22] J. Tuorila, M. Partanen, T. Ala-Nissila, and M. Möttönen, "Efficient protocol for qubit initialization with a tunable environment," *npj Quantum Information*, vol. 3, no. 1, pp. 1–12, 2017.

[23] A. Opremcak, I. Pechenezhskiy, C. Howington, B. Christensen, M. Beck, E. Leonard, J. Suttle, C. Wilen, K. Nesterov, G. Ribeill *et al.*, "Measurement of a superconducting qubit with a microwave photon counter," *Science*, vol. 361, no. 6408, pp. 1239–1242, 2018.

[24] J. Kawahara, T. Saitoh, and R. Yoshinaka, "The time complexity of the token swapping problem and its parallel variants," in *International Workshop on Algorithms and Computation*. Springer, 2017, pp. 448–459.

[25] A. Zulehner, A. Paler, and R. Wille, "Efficient mapping of quantum circuits to the ibm qx architectures. in 2018 design, automation test in europe conference exhibition (date)," 2018.

[26] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. Pereira, "Qubit allocation as a combination of subgraph isomorphism and token swapping," *Proceedings of the ACM on Programming Languages*, vol. 3, no. OOPSLA, pp. 1–29, 2019.
[27] A. M. Childs, E. Schoute, and C. M. Unsal, "Circuit transformations for quantum architectures," *arXiv preprint arXiv:1902.09102*, 2019.
[28] L. Burgholzer, S. Schneider, and R. Wille, "Limiting the search space in optimal quantum circuit mapping," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2022, pp. 466–471.
[29] M. G. Pozzi, S. J. Herbert, A. Sengupta, and R. D. Mullins, "Using reinforcement learning to perform qubit routing in quantum compilers," *arXiv preprint arXiv:2007.15957*, 2020.
[30] R. Wille, L. Burgholzer, and A. Zulehner, "Mapping quantum circuits to ibm qx architectures using the minimal number of swap and h operations," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.
[31] A. Sinha, U. Azad, and H. Singh, "Qubit routing using graph neural network aided monte carlo tree search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, 2022, pp. 9935–9943.
[32] L. Lao, H. van Someren, I. Ashraf, and C. G. Almudever, "Timing and resource-aware mapping of quantum circuits to superconducting processors," *arXiv preprint arXiv:1908.04226*, 2019.
[33] P. Murali, D. C. McKay, M. Martonosi, and A. Javadi-Abhari, "Software mitigation of crosstalk on noisy intermediate-scale quantum computers," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2020, pp. 1001–1016.
[34] C.-Y. Huang, C.-H. Lien, and W.-K. Mak, "Reinforcement learning and dear framework for solving the qubit mapping problem," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.
[35] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, "Simulating large quantum circuits on a small quantum computer," *Physical review letters*, vol. 125, no. 15, p. 150504, 2020.
[36] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, "Cutqc: using small quantum computers for large quantum circuit evaluations," in *Proceedings of the 26th ACM International conference on architectural support for programming languages and operating systems*, 2021, pp. 473–486.
[37] F. Hua, Y. Chen, Y. Jin, C. Zhang, A. Hayes, Y. Zhang, and E. Z. Zhang, "Autobraid: A framework for enabling efficient surface code communication in quantum computing," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 925–936.
[38] M. Beverland, V. Kliuchnikov, and E. Schoute, "Surface code compilation via edge-disjoint paths," *PRX Quantum*, vol. 3, no. 2, p. 020342, 2022.
[39] M. Zhu, H. Fu, J. Wu, C. Zhang, W. Xie, and X.-Y. Li, "Ecmas: Efficient circuit mapping and scheduling for surface code," in *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2024, pp. 158–169.
[40] A. Molavi, A. Xu, M. Diges, L. Pick, S. Tannu, and A. Albarghouthi, "Qubit mapping and routing via maxsat," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022, pp. 1078–1091.
[41] A. Jüttner and P. Madarasi, "Vf2++—an improved subgraph isomorphism algorithm," *Discrete Applied Mathematics*, vol. 242, pp. 69–81, 2018.
[42] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "Qasmbench: A low-level qasm benchmark suite for nisq evaluation and simulation," *arXiv preprint arXiv:2005.13018*, 2020.
[43] B. Tan and J. Cong, "Optimality study of existing quantum computing layout synthesis tools," *IEEE Transactions on Computers*, vol. 70, no. 9, pp. 1363–1373, 2020.

Hao Fu received his Bachelor's degree in Software Engineering from Fuzhou University, China, in 2017. He is currently pursuing his Ph.D. at the University of Science and Technology of China. His research interests include quantum architecture, quantum circuit optimization, and quantum-classical hybrid systems.

Mingzheng Zhu received her B.S. degree in Information and Software Engineering from the University of Electronic Science and Technology of China, Sichuan, China, in 2019. She is currently pursuing a Ph.D. in Computer Science and Technology at the University of Science and Technology of China, Hefei. Her research interests focus on quantum error correction and quantum architecture.

Fangzheng Chen received his B.S. degree in Mechatronic Engineering from the Suzhou University, Suzhou, China, in 2022. He is currently a master candidate at the School of Computer Science and Technology, University of Science and Technology of China, Hefei. His research interests include quantum algorithms and quantum information theory.

Chi Zhang is an associate professor in Hefei University of Technology. He received his B.S. degree in Computer Science and Technology from the University of Science and Technology of China (USTC) with the honor of The Talent Program in Computer and Information Science and Technology in 2017 and got his Ph.D. degree in Computer Science and Technology from USTC in 2023. He is currently an associate professor at Hefei University of Technology. His main research interests are data center networking, cloud computing, and algorithms.

Jun Wu received his B.S. degree in Information and Computing Science from the Hefei University of Technology, Hefei, China, in 2019. He is currently a Ph.D. candidate at the School of Computer Science and Technology, University of Science and Technology of China, Hefei. His research interests include quantum algorithms and quantum information theory.

Wei Xie is an Associate Researcher. He obtained his Bachelor's degree from Nanjing University, Master's degree from Tsinghua University, and Ph.D. from the University of Technology Sydney. During his Ph.D. studies, he was awarded the UTS President's Scholarship (UTSP) and the International Research Scholarship. In July 2020, he joined the School of Computer Science and Technology at the University of Science and Technology of China. His research interests primarily focus on quantum computing and quantum information. He has published several papers in renowned journals and conferences such as IEEE TIT, PRA, Conference on QIP, QIC, IEEE ISIT, and AQIS.

Xiang-Yang Li (Fellow, IEEE/ACM) is a full professor and executive dean at the School of Computer Science and Technology, USTC, Hefei, China. He was a full professor at Illinois Institute of Technology, Chicago, USA. He is an ACM/IEEE Fellow and an ACM Distinguished Scientist. Dr. Li received an MS (2000) and a PhD (2001) degree at the Department of Computer Science from the University of Illinois at Urbana-Champaign, a Bachelor's degree at the Department of Computer Science, and a Bachelor's degree at the Department of Business Management from Tsinghua University, both in 1995. His research spans Artificial Intelligent Internet of Things, mobile computing, data sharing and trading, and privacy. He published a monograph "Wireless Ad Hoc and Sensor Networks: Theory and Applications".