Dynamic Quantum Circuit Compilation

Kun Fang, *Member, IEEE*, Munan Zhang, Ruqi Shi, and Yinan Li

***Abstract*—The practical applications of quantum computing are currently limited by the small number of available qubits. Recent advances in quantum hardware have introduced mid-circuit measurements and resets, enabling the reuse of measured qubits and thus reducing the qubit requirements for executing quantum algorithms. In this work, we present a systematic study of dynamic quantum circuit compilation, a process that transforms static quantum circuits into their dynamic equivalents with fewer qubits through qubit reuse. We establish the first graph-based framework for optimizing qubit-reuse compilation. In particular, we characterize the task of finding the optimal compilation strategy for maximizing qubit reuse using binary integer programming and provide efficient heuristic algorithms for devising general compilation strategies. We conduct a thorough analysis of quantum circuits with practical relevance and offer their optimal qubit-reuse compilation strategies. We also perform a comparative analysis against state-of-the-art approaches, demonstrating the superior performance of our methods in both structured and random quantum circuits. Our framework lays a rigorous foundation for understanding dynamic quantum circuit compilation via qubit reuse, holding significant promise for the practical implementation of large-scale quantum algorithms on quantum computers with limited resources.**

***Index Terms*—Quantum circuit compilation, dynamic circuit, compiler optimization, code sequence optimization, qubit reuse.**

Received 21 November 2024; revised 19 September 2025; accepted 7 December 2025. Date of publication 15 December 2025; date of current version 15 January 2026. Kun Fang was supported in part by the National Natural Science Foundation of China under Grant 92470113 and Grant 12404569; in part by Shenzhen Science and Technology Program under Grant JCYJ20240813113519025; in part by Shenzhen Fundamental Research Program under Grant JCYJ20241202124023031; in part by the General R&D Projects of 1 + 1+1 CUHK-CUHK(SZ)-GDST Joint Collaboration Fund under Grant GRDP2025-022; and in part by the University Development Fund under Grant UDF01003565. Yinan Li was supported in part by the National Key Research and Development Program of China under Grant 2024YFE0102500; in part by the National Nature Science Foundation of China under Grant 62302346; in part by Hubei Provincial Nature Science Foundation of China under Grant 2024AFA045; and in part by the Fundamental Research Funds for the Central Universities under Grant 2042025kf0023. Recommended for acceptance by A. K. Jones. *(Corresponding authors: Kun Fang; Yinan Li.)*

Kun Fang and Munan Zhang are with the School of Data Science, The Chinese University of Hong Kong, Shenzhen, Guangdong 518172, China (e-mail: kunfang@cuhk.edu.cn).

Ruqi Shi is with the Photonics Research Group, Department of Information Technology, Ghent University-IMEC, 9052 Ghent, Belgium.

Yinan Li is with the School of Artificial Intelligence, Wuhan University, National Center for Applied Mathematics in Hubei, and the Hubei Key Laboratory of Computational Science, Wuhan 430072, China, and also with Wuhan Institute of Quantum Technology, Wuhan 430206, China (e-mail: Yinan.Li@whu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TC.2025.3643826, provided by the authors.

Digital Object Identifier 10.1109/TC.2025.3643826

I. Introduction

QUANTUM computing has emerged as a promising avenue for solving intractable problems that are beyond the reach of classical computing, such as integer factoring [1], large database search [2], chemistry simulation [3] and machine learning [4]. To put the theoretically blueprinted quantum advantages into use, it is essential to execute them on real-world quantum computers, moving beyond theoretical concepts and simulation environments. This requires the process of quantum circuit compilation, which optimizes resource utilization and translates high-level quantum algorithms into low-level quantum instructions suitable for target hardware. The compilation process typically involves refining the gate structure of quantum circuits to enhance result fidelity, such as reducing the entangling gate count by simplifying Clifford subcircuits [5], eliminating redundant gates [6], [7], and optimizing T-gate usage [8], [9]. Additionally, it also includes mapping logical circuits to specific quantum device architectures [10], [11] while minimizing the insertion of SWAP gates [12], [13], [14] and reducing circuit depth [15], [16].

Previous studies on quantum circuit compilation have primarily focused on the manipulation of *static quantum circuits*, where computation is carried out on an initially prepared quantum state, and measurements are applied only at the end of the circuit to extract the results. However, recent advancements in quantum hardware have enabled a more versatile paradigm, where measurements and qubit resets can occur during the execution of the quantum circuit. Furthermore, these circuits enable real-time classical information to be fed forward, allowing future instructions to be controlled based on prior measurement outcomes [17], [18]. This emerging paradigm, characterized by the ability to dynamically adjust the circuit during execution, is referred to as *dynamic quantum circuits*. It plays a crucial role in advancing recent breakthroughs in quantum error correction [19], [20], [21].

In this work, we investigate *dynamic quantum circuit compilation*, a new procedure within the quantum circuit compilation pipeline that rewires static quantum circuits into equivalent dynamic circuits with fewer qubits. An illustrative example is shown in Fig. 1, where a 3-qubit static quantum circuit is compiled into a 2-qubit dynamic circuit by recycling the first qubit after its measurement and subsequently reusing it for operations on the third qubit of the original circuit. This type of process focuses on reordering instructions and reallocating qubit registers, while preserving both the number and type of circuit operations. It complements other circuit optimization techniques and can be seamlessly integrated into existing

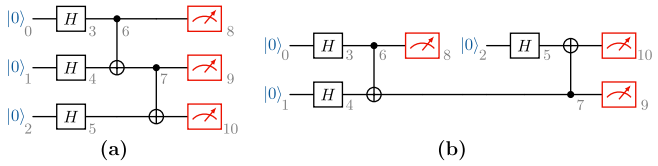

Fig. 1. An example of dynamic quantum circuit compilation from (a) to (b). (a) A static quantum circuit with 3 qubits. (b) An equivalent dynamic quantum circuit with 2 qubits.

compilation workflows. For example, it can be applied after the removal of redundant gates or prior to mapping the circuit onto a specific quantum architecture.

A closely related problem in classical compiler optimization is minimum register instruction sequencing (MRIS) [22], which seeks to find an instruction schedule that minimizes the number of registers used in classical computing architectures. The concept of qubit reuse in quantum circuit compilation was first introduced in [23], which employed the notion of wire recycling applied to predefined ancilla qubits. Subsequent work in [24] developed a compiler-assisted tool and exploited the trade-off between qubit reuse, fidelity, gate count, and circuit duration. More recently, [25] introduce an SAT-based model for optimizing qubit-reuse on near-term quantum devices of small size. The work in [26] investigated qubit-reuse compilation using the causal structure of the circuits and provided heuristic algorithms for general compilation.

In this work, we propose the first graph-based dynamic quantum circuit compilation framework, aimed at minimizing the number of qubits required to implement a circuit. Our main contributions can be summarized as follows:

- The first general framework for optimizing dynamic circuit compilation through graph manipulation and a rigorous mathematical model for optimal compilation in terms of qubit-reuse. Our framework primarily targets qubit savings but is also adaptable to other scenarios, such as optimizing tradeoffs among circuit width, depth, and related factors.
- Efficient approach to determine whether a given static quantum circuit can be reduced to a smaller circuit through qubit-reuse, coupled with a solver that leverages multiple heuristic algorithms for designing dynamic circuit compilation strategies in general.
- Compilation of quantum circuits with commutable structures, thereby addressing an open challenge for circuit compilation highlighted in the prior study [26]. This holds particular significance for practical applications in quantum machine learning [27], measurement-based quantum computation [28], and the pursuit of quantum supremacy [29].
- Qubit-reuse-optimal compilation strategies for quantum circuits of practical relevance, including well-known quantum algorithms in quantum computation, ansatzs in quantum machine learning, and measurement-based quantum computation crucial for quantum network. These optimal compilation strategies establish tight lower bounds and serve as benchmarks for other variants of qubit reuse compilation methods.
- Numerical evaluations of our heuristic algorithms on both structured and random quantum circuits, highlighting the superior performance of our methods over the state-of-the-art results by [26]. In particular, experiments show that our approach outperforms in approximately 98.5% of randomly generated quantum circuits and nearly 100% of randomly generated instantaneous quantum polynomial (IQP) circuits.
- Noisy simulations conducted with experimental parameters from a real-world trapped-ion quantum computer also demonstrate a significant improvement in qubit reduction and circuit fidelity, confirming the effectiveness of the compilation process.

Our framework can be viewed as the quantum analog of MRIS and is well-motivated by the following key factors: 1) Previous studies have mainly focused on superconducting quantum computers [24], [25], which suffer from limited qubit connectivity and short coherence time. In contrast, trapped-ion quantum systems offer all-to-all connectivity and exceptionally long coherence times. However, scalability remains a significant bottleneck in their development, underscoring the urgent need to minimize qubit usage for enabling scalable quantum computing on these platforms. 2) Qubit reuse effectively compresses the quantum circuit topology, reducing the need for SWAP gates when mapping the logical circuit to quantum hardware in the later stages of the compilation pipeline. This leads to a final circuit with reduced depth. 3) Additionally, by reducing the number of qubits, we can avoid the allocation of faulty qubits and links with higher error rates, ultimately improving the fidelity and performance of the compiled circuit [24], [25]. It is worth stressing that, although our motivation for minimizing circuit width originates from trapped-ion quantum computers, the proposed framework operates at the logical-circuit level, i.e., before decomposition into hardware-native gates, and is therefore architecture-agnostic, making it applicable to a variety of quantum hardware platforms. While our primary goal is to minimize qubit usage, the framework is also flexible enough to accommodate other scenarios, such as exploring trade-offs between circuit width, depth, and related factors, which may be of importance on other platforms.

The rest of this work is structured as follows: Section II and Section III cover the notations and fundamental concepts used. Section IV establishes the core principles behind dynamic quantum circuit compilation through graph manipulation. Section V introduces an efficient approach for determining the reducibility of a static quantum circuit. Section VI formulates dynamic quantum circuit compilation for maximizing qubit reuse as a binary integer programming problem. Section VII presents our solver and heuristic algorithms. Section VIII evaluates the proposed methods. Section IX reviews some related works. Section X concludes our study.

II. Preliminaries

Notation: We denote I_n as the $n \times n$ identity matrix, J_n as the $n \times n$ all-one matrix, and O_n as the $n \times n$ zero matrix.

We adopt the convention that the row and column indices of matrices start from zero.

Directed Graph: A graph is represented by an ordered pair $G = (V, E)$, where V and E denotes the set of vertices and edges, respectively. In a directed graph, each edge (u, v) is an ordered pair where u is the tail and v is the head. The number of edges incident into a vertex v is called its *indegree* and denoted as $\delta^-(v)$. The number of edges incident out of a vertex v is called its *outdegree* and denoted as $\delta^+(v)$. A vertex with zero indegree is called a root and a vertex with zero outdegree is called a terminal. A vertex v is *reachable* from another vertex u if there exists a direct path from u to v in the graph. A *directed acyclic graph* (DAG) is a directed graph with no directed cycles. A *topological ordering* of a DAG is a linear ordering of its vertices such that for every directed edge (u, v), vertex u occurs before vertex v in the ordering. A *bipartite graph* is a graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in set U to a vertex in set V. We use $G = (U, V, E)$ to denote a bipartite graph with parts U, V and edges E. A bipartite graph is complete if every vertex in U is connected to every vertex in V.

Matrix Representation of Graph: Let $G = (V, E)$ be a directed graph where $V = \{v_0, \cdots, v_{k-1}\}$. Its *adjacency matrix* is a Boolean matrix, denoted as $A(G)$, of size $k \times k$ whose (i, j)-th entry is one if the directed edge $(v_i, v_j) \in E$ and zero otherwise. If a bipartite graph $G = (U, V, E)$ with $U = \{u_0, \cdots, u_{k-1}\}$, $V = \{v_0, \cdots, v_{k-1}\}$ and all edges pointing from U to V, then its adjacency matrix can be written as $A(G) = \left(\begin{smallmatrix} O_k & X \\ O_k & O_k \end{smallmatrix}\right)$ where X is a $k \times k$ matrix in which $X_{ij} = 1$ if $(u_i, v_j) \in E$. We call this submatrix the *biadjacency matrix* of the bipartite graph and denote it as $B(G)$. The biadjacency matrix of a complete bipartite graph is an all-one matrix. A $k \times k$ matrix M is *nilpotent* if there exists an integer l with $1 \leq l \leq k$ such that $M^l = O_k$. It is well-known that a directed graph is acyclic if and only if its adjacency matrix is nilpotent [30].

III. Quantum Circuit and Its Representations

A. Quantum Circuits and Quantum Instructions

A quantum circuit is a mathematical model to represent quantum computations. It consists of a sequence of quantum operations, including quantum gates, measurements and reset operations. Quantum gates are unitary transformation that manipulate quantum states. Measurements extract classical information from quantum states. After a measurement, a qubit can be reset to a known state (typically $|0\rangle$) and reused for subsequent computation. In this work, we focus on single-qubit and two-qubit gates without loss of generality, noting that most results apply to multi-qubit gates with slight modifications.

Definition 1: A quantum circuit is static if its structure is fully predetermined prior to execution, that is, it does not contain mid-circuit measurements and all measurements are performed only at the end of the circuit. A quantum circuit is dynamic if it contains mid-circuit measurements and possibly feed-forward classical control of quantum gates, which may include reset operations as a special case.

Definition 2: A static quantum circuit $\mathcal{C}$ is considered reducible if it can be written as an equivalent dynamic quantum circuit $\mathcal{C}'$ with fewer qubits. Otherwise, it is irreducible. A quantum circuit and its reduced circuit are equivalent if their measurement outcomes yield identical distributions.

In the context of quantum circuit compilation, quantum circuit instructions are more frequently used, where each quantum operation in the circuit is represented by a unique instruction. An instruction can take one of the following forms: $\mathbf{r}(q)$, which resets qubit q to $|0\rangle$; $g(\boldsymbol{q})$, which applies the unitary gate g to a list of qubits $\boldsymbol{q}$; or $\mathbf{m}(q)$, which measures qubit q. This approach allows a quantum circuit to be expressed as a sequential list of instructions, arranged according to their intended execution order. For example, the static circuit depicted in Fig. 1(a) can be represented as $\mathcal{C} = [\mathbf{r}(0), \mathbf{r}(1), \mathbf{r}(2), H(0), H(1), H(2), CX(0, 1), CX(1, 2), \mathbf{m}(0), \mathbf{m}(1), \mathbf{m}(2)]$. Throughout this work, we do not restrict on any specific set of universal gates, as dynamic quantum circuit compilation preserves both the type and number of gates.

B. Graph Representation of Quantum circuits

Given that a quantum circuit essentially constitutes an ordered sequence of quantum instructions, it is natural to employ a directed graph to represent the causal relationships among them. In this context, the directed graph effectively preserves the execution constraints of quantum operations.

Definition 3: The DAG representation of a quantum circuit is a directed graph constructed as follows: each quantum instruction indexed by i corresponds to a vertex i in the graph, and a directed edge from vertex u to vertex v indicates that the instruction associated with u must precede the instruction associated with v.

In many instances of interest, the original circuit is not unique and can be represented with different ordering of gates. A typical example is given by the instantaneous quantum polynomial (IQP) circuits [29], which takes the form of $H^{\otimes n} D H^{\otimes n}$. Here, H denotes the Hadamard gate, and D constitutes a block of gates that are diagonal in the computational basis and consequently can be applied in any temporal order.

These circuits do not have pre-determined structures, and imposing any dependencies among commuting gates may limit the opportunities for qubit-reuse. To handle circuits with commutable structure, we avoid imposing dependencies between commutable gates and only establish edges between operations with pre-defined ordering when generating the DAG representation of the circuit.

In this work, we employ the DAG representation to investigate the dynamic quantum circuit compilation problem. To facilitate our analysis, all vertices within the DAG representation are categorized into three distinct groups: (1) **roots** R, vertices with zero indegree, which correspond to the first layer of quantum operations on each qubit (typically reset operations), (2) **terminals** T, vertices with zero outdegree, which correspond to the last layer of quantum operations on each qubit (typically

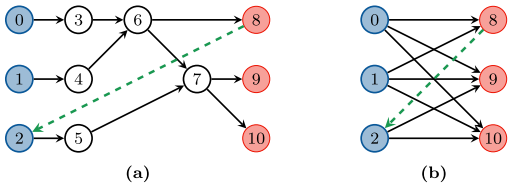

Fig. 2. An illustration of dynamic circuit compilation via graph manipulation. Root, internal and terminal vertices are marked in blue, white and red, respectively. The added edge is marked in green. (a) DAG representation with added edge. (b) Simplified DAG representation with added edge.

quantum measurements), and (3) **internal vertices** I, vertices with nonzero indegree and outdegree, which correspond to the intermediate quantum gates. Therefore, the DAG representation of a quantum circuit can be denoted as $G(R \cup I \cup T, E)$. In the following discussion, we assume that each qubit in a static quantum circuit start from a reset operation and end with a measurement. This implies that the circuit width is equal to the number of root (or terminal) vertices within the DAG representation. For an input static quantum circuit with n qubits and m instructions, the time complexity of the algorithm for constructing its DAG representation is $O(m^2/n)$.

Regarding our compilation problem, all essential information resides within the reachability from roots to terminals within the DAG representation, while the internal vertices serve to facilitate and transmit this reachability. Consequently, we can further streamline the DAG representation and concentrate on the *simplified DAG representation* of the quantum circuit. Fig. 2 showcases the DAG and simplified DAG representations of the static circuit in Fig. 1(a).

Definition 4: The simplified DAG representation of a quantum circuit $G_s(R, T, E_s)$ is a bipartite graph, with R and T representing the sets of roots and terminals in the DAG representation G. An edge $(r, t) \in E_s$ connects a root $r \in R$ to a terminal $t \in T$ if a directed path exists from r to t within the DAG representation.

IV. Dynamic Quantum Circuit Compilation via Graph Manipulation

In this section, we present the mathematical formulation of the quantum circuit compilation problem using its graph representation, which serves as a pivotal foundation for subsequent in-depth investigations. Drawing inspiration from the example illustrated in Fig. 1, dynamic quantum circuit compilation via qubit-reuse entails deferring the reset operation on one qubit until after the measurement of another qubit. In the DAG representation, this corresponds to the addition of a directed edge from a terminal to a root, signifying that the corresponding reset operation occurs after the measurement has been executed.

For instance, Fig. 2(a) provides a DAG representation of the static circuit depicted in Fig. 1(a). The qubit-reuse in Fig. 1 is depicted by the addition of a new edge (marked as a dashed green line) from terminal 8 to root 2. With the inclusion of this new edge, the resulting DAG exactly mirrors the DAG representation of the dynamic quantum circuit depicted in Fig. 1(b). This new edge can also be integrated into the simplified DAG representation as shown in Fig. 2(b).

With this in mind, we have the following result.

Theorem 1: Let $G = (R \cup I \cup T, E)$ be the DAG representation of a static quantum circuit $\mathcal{C}$. Then compiling the quantum circuit via qubit-reuse is equivalent to adding edges to G (Let E' be the set of added edges) such that

- $\forall\, (t, r) \in E'$, it has $t \in T$ and $r \in R$;
- $\forall\, (t, r) \in E'$, it has $\delta^+(t) = 1$ and $\delta^-(r) = 1$;
- $G' = (R \cup I \cup T, E \cup E')$ is an acyclic graph,

where δ^+, δ^- denote outdegree and indegree, respectively. The new graph G' is referred to as the modified DAG.

Proof: Dynamic quantum circuit compilation through qubit-reuse can be illustrated by the addition of directed edges from terminals to roots in the DAG representation. These added edges must satisfy the following conditions:

1) Resetting a qubit is only feasible if all preceding operations are completed. So added edges should start from terminals. After resetting, the qubit can only be reused to execute operations on another qubit from the beginning. Since the circuit width is determined by the number of roots in the DAG, added edges should end at roots to reduce circuit width.
2) A reused qubit can only accommodate one reset operation. So the added edges should have no common tails. Moreover, since the circuit width is determined by the number of roots in the DAG representation, the added edges having common heads will not help. Thus, all added edges should not have overlapping vertices.
3) Since directed edges represent the execution constraint of operations, any cycle in the graph would imply a dependency of past operations on future ones, which violates the causal relation. Therefore, the addition of these directed edges must not introduce any cycles to ensure the compiled circuit is well-defined.

Conversely, any graph manipulation that complies with these constraints corresponds to a valid dynamic circuit compilation strategy. Given that the DAG representation preserves the execution order of quantum operations within the quantum circuit, the topological sorting on the modified DAG establishes a viable execution sequence for rearranging the circuit instructions list. Since our actions solely involve rewiring the original quantum circuit, the resulting compiled circuit maintains its equivalence to the static circuit. It's worth noting that all specified conditions apply regardless of whether we are working with a DAG or a simplified DAG representation. □

With the above understanding of our graph-based approach, we are ready to introduce the structure of our compilation framework in Fig. 3, where each box is a component of the framework and the labeled dashed arrows represent the data flow between them. The compilation process is directed by the blue arrows. For an input static circuit $\mathcal{C}$, we begin by constructing its DAG representation G. The Reducibility module employs Depth-First Search (DFS) on G to determine the reducibility of the input circuit. It also translates the graph manipulation problem to a binary integer programming (BIP) problem. Next, the Heuristic module applies heuristic

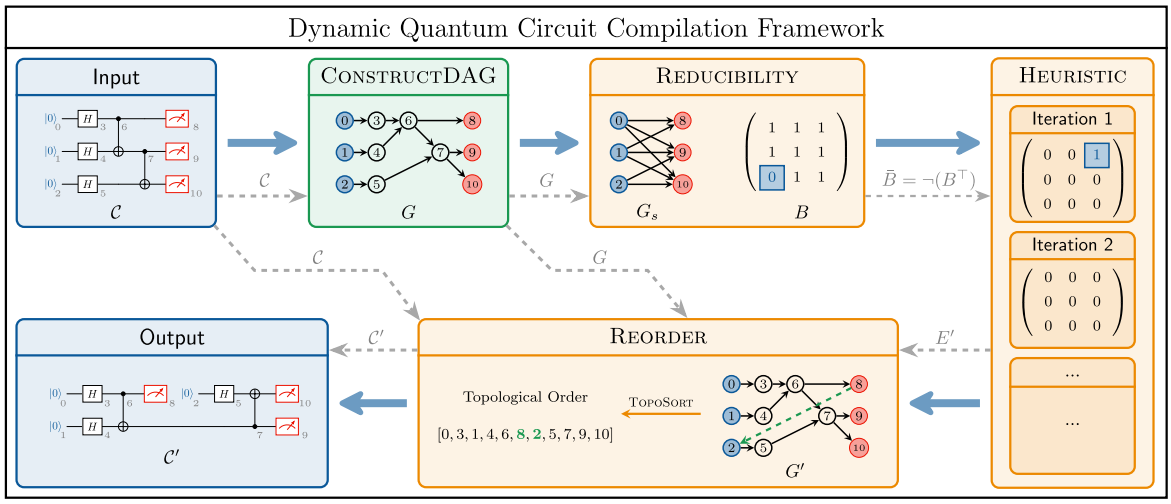

Fig. 3. Overview of the proposed dynamic quantum circuit compilation framework. Each box represents a component, and dashed arrows indicate data flow, with blue arrows highlighting the main workflow. Starting from an input static circuit $\mathcal{C}$, its DAG representation G is first constructed. The REDUCIBILITY module analyzes G to determine circuit reducibility and formulates the problem as a binary integer programming (BIP) task. The HEURISTIC module then approximates the BIP solution using heuristic algorithms, producing a set of feasible edge additions E'. these edges are integrated into G by the reorder module, which performs a topological sort (TopoSort) to establish a valid execution order and reallocates qubits according to the reuse strategy, resulting in the compiled dynamic circuit $\mathcal{C}'$.

algorithms to find an approximate solution to the BIP problem, which is then transformed into a feasible edge addition strategy, E'. The Reorder module incorporates all edges in E' into G and performs a topological sort (TopoSort) on the modified DAG G' to establish a viable execution sequence of quantum instructions. Finally, the Reorder module reallocates the qubit register based on the reuse strategy and outputs the compiled dynamic circuit $\mathcal{C}'$.

V. Determine the Reducibility

Prior to the actual compilation for a static circuit, we shall first determine if it can be reduced at all. This is clear from Theorem 1 that a quantum circuit is reducible if and only if we can add at least one edge to its DAG while adhering to the specified conditions.

Proposition 1: A static quantum circuit is irreducible if and only if its simplified DAG is a complete bipartite graph.

Proof: If the simplified DAG is a complete bipartite graph, then any terminal is reachable from any root. Adding any additional edge from a terminal to a root in such a graph will inevitably create a directed cycle. Consequently, the circuit width can not be reduced through qubit reuse. Conversely, if the simplified DAG is not complete, it implies that there exists at least one terminal t that is not reachable from some root r. In this situation, adding the directed edge (t, r) to the graph does not introduce any cycles. Thereby, the corresponding circuit can be reduced to a circuit with a smaller width. □

This result shows that the reducibility of a static quantum circuit can be determined by checking if the biadjacency matrix of the simplified DAG is an all-one matrix. For this, we can exploit Depth-First Search (DFS) algorithm to explore the reachability between roots to terminals in a given DAG. For a static quantum circuit with n qubits and m instructions, the worst-case time complexity of this approach is $O(mn)$.

VI. Qubit-Reuse – Optimal Quantum Circuit Compilation

Theorem 1 establishes a one-to-one correspondence between a graph manipulation scheme and a dynamic quantum circuit compilation strategy. Therefore, our primary aim of *minimizing* the number of qubits is equivalent to *maximizing* the number of added edges. In this section, we formulate the task of finding optimal compilation strategies for maximizing qubit reuse as an binary integer programming (BIP) problem. This model provides a *theoretical* characterization of qubit-reuse–optimal compilation and serves as the foundation for devising solver and heuristic algorithms (Section VII) and analyzing qubit-reuse–optimal compilation strategies for several quantum circuits (Section VIII).

Note that Theorem 1 remains valid when working on the simplified DAG. Let $G_s(R, T, E_s)$ denote the simplified DAG of a static circuit with n qubits, where $R = \{r_0, ..., r_{n-1}\}$ and $T = \{t_0, ..., t_{n-1}\}$. Its adjacency matrix $A(G)$ is a $2n \times 2n$ block anti-diagonal matrix, where the first n rows/columns correspond to n roots, and the last n rows/columns correspond to n terminals. Since all edges in $G_s(R, T, E_s)$ direct from roots to terminals, only the $n \times n$ submatrix in the upper right corner contains non-zero entries. Therefore, its adjacency matrix can be written as $A(G_s) = \left(\begin{smallmatrix} O_n & B \\ O_n & O_n \end{smallmatrix}\right)$, where B is the biadjacency matrix of G_s.

Let E' be the set of edges added to G_s. We represent these edges with an $n \times n$ matrix F, where the entry $F_{ij} = 1$ if a directed edge $(t_i, r_j) \in E'$ and 0 otherwise. Our objective of maximizing the number of added edges is equivalent to maximizing the objective function

$$\sum_{i,j=0}^{n-1} F_{ij}. \tag{1}$$

Next, we translate the constraints stated in Theorem 1 into the constraints on matrix F. A key observation is that the absence of a directed edge from r_i to t_j in G_s allows the addition of a candidate edge from t_j to r_i. All these candidate edges can be represented by an $n \times n$ matrix $\bar{B}$, referred to as *the candidate matrix*. In fact, the candidate matrix can be efficiently computed as $\bar{B} = \neg (B^\top)$, where $\neg$ denotes the logical NOT. Another observation is that each added edge should be selected from the set of all candidate edges, which can be expressed as

$$F_{ij} \leq \bar{B}_{ij}, \text{ for all } i, j \in \{0, 1, 2, ..., n-1\}. \tag{2}$$

The second constraint in Theorem 1 requires that all added edges should not share common vertices, which implies that

$$\sum_{j=1}^{n-1} F_{ij} \leq 1, \text{ for all } i \in \{0, 1, 2, ..., n-1\}, \tag{3}$$

$$\sum_{i=1}^{n-1} F_{ij} \leq 1, \text{ for all } j \in \{0, 1, 2, ..., n-1\}. \tag{4}$$

Finally, after adding edges in E', the resulting graph $G'_s(R, T, E \cup E')$ should remain acyclic. According to [30], this requirement is equivalent to the adjacency matrix of G'_s being nilpotent. That is,

$$A(G'_s) = \begin{pmatrix} O_n & B \\ F & O_n \end{pmatrix} \text{ nilpotent.} \tag{5}$$

Therefore, the BIP problem for qubit-reuse-optimal circuit compilation is to maximize the objective function (1) subject to constraints (2) to (5). While the constraint (2) is indeed implied by the nilpotency constraint (5), we explicitly state it as it proves helpful in the design of solver in Section VII and in the analysis of optimal compilation strategies in Section VIII. The difficulty in solving the binary integer programming arises from the presence of the nilpotent constrain, which is non-convex and non-linear and thereby can not be directly encoded in standard BIP solver (e.g. Gurobi [31]). While the BIP is not solved directly in our numerical experiments, it provides a rigorous theoretical framework that guides solver design and supports the derivation of qubit-reuse–optimal strategies.

VII. Algorithms Design

The previous section demonstrated that the dynamic quantum circuit compilation problem is essentially a BIP problem with n^2 Boolean variables and non-linear conditions. While checking the reducibility of a quantum circuit is a polynomial-time task, finding the optimal qubit-reuse compilation strategy could potentially require exponential time. In this section, we propose efficient algorithms for approximate solutions.

A. Solving Procedure

The BIP problem seeks to select the maximum number of candidate edges from the candidate matrix that can be added to the DAG representation while adhering to three constraints. Our solver (Algorithm 1) implements this idea iteratively: starting from the candidate matrix $\bar{B}$, it identifies one candidate edge based on a heuristic strategy h, and then updates the candidate matrix to ensure that all edges considered in the subsequent steps remain compliant with the specified constraints (Algorithm 2). Specifically, before adding the edge (t_i, r_j), let set R_i represents all roots capable of reaching terminal t_i, and set T_j represents all terminals reachable from root r_j. Adding the edge (t_i, r_j) allows any root in R_i to reach any terminal in T_j, thereby invalidating all edges $(t, r) \in T_j \times R_i$. To prevent shared vertices among added edges, all entries in the t_i row and the r_j column of the candidate matrix are set to zero. The algorithm then proceeds to the next iteration until the candidate matrix is reduced to a zero matrix.

Algorithm 1 SOLVER

Input: candidate matrix $\bar{B}$, heuristic strategy h
Output: added edges E'

```
Initialize E' = ∅
while B̄ ≠ O_n do
    (t_i, r_j) = h(B̄)
    E' = E' ∪ {(t_i, r_j)}
    B̄ = UpdateCandidate(B̄, (t_i, r_j))
end while
return E'
```

Algorithm 2 UPDATECANDIDATE

Input: candidate matrix $\bar{B}$, added edge (t_i, r_j)
Output: updated candidate matrix after adding (t_i, r_j)

```
R_i = {r_k | B̄[t_i, r_k] = 0}
T_j = {t_k | B̄[t_k, r_j] = 0}
for each (t, r) ∈ T_j × R_i do
    B[t, r] = 0
end for
B̄[t_i, :] = 0, B̄[:, r_j] = 0
return B̄
```

Fig. 4 demonstrates the solving procedure of our algorithm for a 5-qubit Bernstein-Vazarini circuit. Panel (a) shows the biadjacency matrix B of the input circuit with row and column indices, and panel (b) shows the corresponding candidate matrix $\bar{B}$. Suppose the selected edge is (t_3, r_1), highlighted with a green box. Adding this edge enables any root in $R_3 = \{r_0, r_2, r_3, r_4\}$ (rows with a 1 in the t_3 column, marked with blue boxes in panel(a)) to reach any terminal in $T_1 = \{t_1\}$ (columns with a 1 in the r_1 row, marked with a red box in panel (a)). Consequently, we need to update all entries in $T_1 \times R_3 = \{(t_1, r_0), (t_1, r_2), (t_1, r_3), (t_1, r_4)\}$ to zero, as shown in panel (c). Subsequently, all entries in the t_3 row and the r_1 column are set to zero, as depicted in panel (d), and the algorithm proceeds to the next iteration with the updated candidate matrix. For completeness, the reference circuit of the 5-qubit Bernstein–Vazirani algorithm, together with its DAG and simplified DAG, is provided in Appendix B of the Supplemental Material.

Although our algorithm primarily aims to minimize the circuit width, it retains the flexibility to halt the main loop based

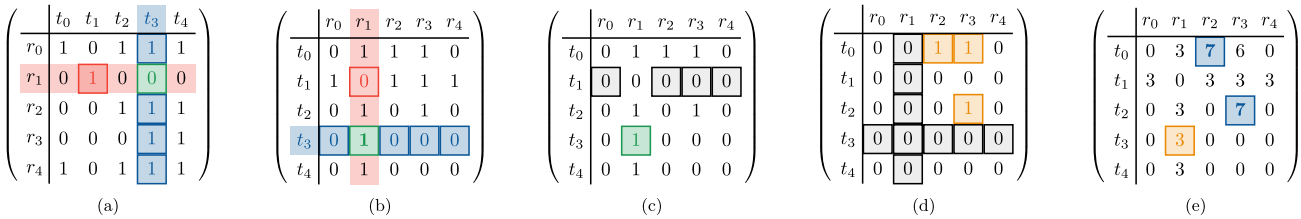

Fig. 4. An illustration of the solving procedure for a 5-qubit Bernstein–Vazirani circuit. (a) Biadjacency matrix B of the input circuit. (b) Candidate matrix $\bar{B} = \neg(B^\top)$, with the selected candidate edge (t_3, r_1) highlighted in a green box. (c) After adding (t_3, r_1), all entries in $T_1 \times R_3 = \{(t_1, r_0), (t_1, r_2), (t_1, r_3), (t_1, r_4)\}$ (gray boxes) are set to zero. (d) All entries in the t_3 row and r_1 column (gray boxes) are set to zero, and the algorithm proceeds to the next iteration with the updated candidate matrix. (e) Scores for all candidate edges in the first iteration of the greedy heuristic. Adding edge (t_3, r_1) leaves three 1s in the updated matrix, giving this edge a score of 3.

on specific criteria, such as a user-defined qubit count. This flexibility also allows us to explore the trade-off between circuit width, depth and other related factors (see e.g. experiment in Table II).

B. Heuristics Design

It remains to find an appropriate heuristic strategy to identify candidate edge in each iteration, such that the number of iterations is maximized. We employ the following two heuristics that have shown robust performance in numerical experiments, particularly the greedy approach.

1) Minimum Remaining Values: The Minimum Remaining Values (MRV) heuristic designates the variable with the fewest valid values as the next one for value assignment. For example, in Sudoku solvers, this involves choosing the empty cell with the least potential values. In our context, we first identify the terminal t_i that has the least number of candidate roots, which corresponds the row in the candidate matrix with fewest 1s. Then we connect t_i to one of its candidate roots who has the fewest candidate terminals, which corresponds to the r_j-th column where $r_j = \arg\min_{\bar{B}_{t_i r_j}=1} \sum_{k=1}^{n} \bar{B}_{t_k r_j}$. Note that the role of roots and terminals can be exchanged. In practice, we run both approaches and return the better result. For an input candidate matrix with size $n \times n$, the MRV algorithm has a worst-case time complexity of $O(n^3)$. To further enhance the performance of the heuristic strategy, we introduce a hybrid approach that combines the MRV heuristic with brute-force search. A detailed description and numerical evaluation of this hybrid method are provided in Appendix C of the Supplemental Material.

2) Greedy: Greedy algorithms are time-efficient heuristic strategies that make a locally optimal choice at each iterative step. In our problem, the number of 1s in the candidate matrix represents an upper bound on the iteration numbers of the algorithm. Therefore, we may choose the candidate edge that maximizes the number of remaining candidate edges in the next step, such that the algorithm is likely to step more. That is, we score each edge within the candidate matrix by the number of remaining candidate edges after adding it and select the edge with highest score. For example, Fig. 4(e) reports the scores for all candidate edges in the first iteration of the 5-qubit Bernstein-Vazirani circuit. Adding edge (t_3, r_1) leaves three 1s in the updated matrix (orange boxes in Fig. 4(d)), hence its score is 3, as highlighted in an orange box Fig. 4(e). For an input candidate matrix with size $n \times n$, the greedy algorithm has a worst-case time complexity of $O(n^5)$.

In practice, it is common that multiple candidate edges perform equally. For example, Fig. 4(e) illustrates the scores for each edge in the candidate matrix, where two edges share the highest score 7 (marked in blue). In such scenarios, a deterministic selection (e.g., choosing the one with the smallest index) may inadvertently tie the compilation process to specific qubit labels, potentially restricting algorithm's performance. To overcome this limitation, we randomly choose one among them. This stochastic method leaves the possibility to return enhanced solutions across multiple algorithm runs and has proven to be effective in our numerical experiments.

Remark 1: The scoring rule in the greedy heuristic is flexible and can be replaced with alternative approaches based on specific objectives. One such approach involves evaluating the impact on the circuit depth for adding a candidate edge to the DAG by calculating the length of the critical path (i.e., the longest path in the DAG, which determines the minimum achievable circuit depth). Then the candidate edge can be scored by a predefined cost function that balances circuit width and depth, thereby enabling explicit trade-off evaluation. This formulation essentially encompasses the qubit-saving approach in [24] as a special case.

C. Reorder

After finding a valid edge addition strategy E', we incorporate all edges in E' to G and obtain the modified DAG G'. Then the REORDER algorithm (Algorithm 3) employs a topological sort on G' to establish a feasible execution order, which is utilized to rearrange the compiled circuit instructions list $\mathcal{C}'$. According to our construction of the DAG representation, the instruction corresponding to a vertex v can be accessed via $\mathcal{C}[v]$. For a terminal vertex t, the associated instruction $\mathcal{C}[t]$ is a measurement operation, while for a root vertex r, the associated instruction $\mathcal{C}[r]$ is a reset operation. Let q_t and q_r denote the qubits of the instructions $\mathcal{C}[t]$ and $\mathcal{C}[r]$, respectively. Adding an edge (t, r) indicates that we reuse q_t to execute quantum operations on q_r. Consequently, we can reallocate qubit registers in the compiled circuit $\mathcal{C}'$ according to the derived qubit reuse strategy. For a static quantum circuit with n qubits and m instructions, the time complexity of this algorithm is $O(mn)$.

Algorithm 3 REORDER

Input: Input circuit instructions $\mathcal{C}$, added edges E', DAG representation $G(V, E)$
Output: compiled circuit instructions $\mathcal{C}'$

Initialize $\mathcal{C}' = \emptyset$
$G' = G(V, E \cup E')$
$TopoOrder =$TOPOSORT(G')
for each v in $TopoOrder$ **do**
    $\mathcal{C}'$.append($\mathcal{C}[v]$)
end for
for each $(t, r) \in E'$ **do**
    $Instr, Instr' = \mathcal{C}[t], \mathcal{C}[r]$
    $q_t, q_r = Instr(q), Instr'(q)$
    for each $Instr$ in $\mathcal{C}'$ **do**
        for each q_i in $Instr(\boldsymbol{q})$ **do**
            if $q_i = q_r$ **then** set q_i to q_t
        end for
    end for
end for
return $\mathcal{C}'$

TABLE I
THE MINIMUM NUMBER OF QUBITS REQUIRED TO EXECUTE CERTAIN STRUCTURED QUANTUM CIRCUITS AFTER COMPILATION

Quantum Circuits	Original	Compiled
Bernstein-Vazirani algorithm	$n+1$	2 (1)
Quantum Fourier transform + measurement	n	1
Quantum phase estimation	$n+m$	$n+1$
Shor's algorithm	$3n$	$n+1$
Quantum counting algorithms	$n+m$	$n+1$
Quantum ripple carry adder circuit	n (4)	4 (3)
Linearly entangled circuit with l layers	n	$l+1$
Circularly entangled circuit with l layers	n	3
Pairwise entangled circuit with l layers	n	$2l+1$
Diamond-structured quantum circuit	$2n$	$n+1$
MBQC with cluster state of size (w, d)	wd	$w+1$
MBQC with brickwork state of size (w, d)	wd	$w+1$

VIII. EVALUATIONS

A. Analytical Evaluation

Section VI completely characterizes the optimal quantum circuit compilation for maximizing qubit reuse using binary integer programming. In this section, we conduct a thorough analysis of quantum circuits with practical relevance and provide their optimal qubit reuse compilation schemes. Table I summarizes all reducible quantum circuits studied in this work along with their optimal compilation results.

A few examples in the table can be understood as follows. In the Bernstein-Vazirani algorithm for determining an n-bit secret string s ($n \geq 2$), the circuit uses $n+1$ qubits. The minimum compiled circuit width is 1 for an all-zero string s and 2 otherwise. For the quantum Fourier transform (QFT) on n qubits immediately followed by measurement of all qubits, the computation can be realized using a single-qubit dynamic circuit. Quantum phase estimation (QPE) requires n qubits to accommodate the unitary operation and m additional control qubits. Since the inverse QFT and measurement are used as a subroutine in QPE, the control register can be reduced to a single qubit through dynamic circuit compilation. Shor's algorithm and quantum counting algorithms are special cases of QPE, with $2n$ and m control qubits, respectively, and the same compilation strategy applies. Finally, for the quantum ripple carry adder circuit designed to add two k-bit strings, the circuit requires $n = 3k+1$ qubits ($n \geq 4$). The width-minimized compiled circuit uses 3 qubits when $n = 4$ and 4 qubits when $n > 4$.

Besides the listed examples, we find hat standard quantum circuits implementation for Grover's algorithm and fully entangled circuits are irreducible. It is important to note, however, that a quantum algorithm can often be realized using multiple circuit implementations. Consequently, our analytical results are specific to the particular implementations considered, and it remains possible that alternative implementations of these algorithms could be reducible. Exploring qubit reuse at the algorithmic level is beyond the scope of this work and left for future exploration. In Appendix A of the supplemental material, we detail the specific quantum circuit implementations used in this work, along with their corresponding optimal solutions to the BIP problem. Each proof of optimality involves two parts: one establishes a lower bound by presenting an explicit feasible solution to the integer programming problem, while the other relaxes the optimization by removing the nilpotent condition to establish an upper bound. The optimality of our feasible solution is confirmed by demonstrating that the lower and upper bounds coincide.

B. Numerical Evaluation

In this section, we conduct extensive numerical evaluations of our proposed methods. We begin by analyzing the reducibility of several quantum supremacy circuits. Then we perform a comparative analysis against state-of-the-art approaches on a range of benchmark circuits. We also designed a noisy simulation to demonstrate the practical effectiveness of dynamic circuit compilation in improving circuit fidelity. All numerical evaluation are performed using a self-developed software toolkit, and the code is available on GitHub[1] for reproducibility and verification.

To compare the reducibility of quantum circuits as well as the performance of different compilation methods, we define the *reducibility factor* of a quantum circuit as $r = 1 - n'/n \in [0, 1)$, where n and n' represent the original and the compiled circuit width, respectively. This factor characterizes the extent to which the circuit width can be reduced by a certain algorithm, which is zero if the circuit is irreducible.

1) Quantum Supremacy Circuits: Several quantum experiments, conducted on Sycamore and Zuchongzhi quantum computers, have been used to claim quantum supremacy. The complexity of such circuits is controlled by the number of cycles, which must strike a delicate balance to ensure that deep circuits are executable within the capabilities of current quantum

[1]https://github.com/zhangmunannoe/Dynamic-Quantum-Circuit-Compilation

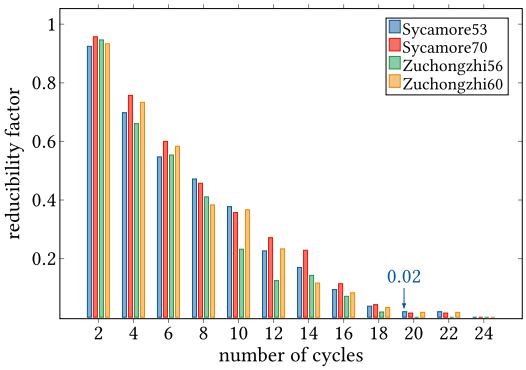

Fig. 5. The reducibility factor of different quantum supremacy circuits with varying number of cycles using the greedy heuristic algorithm.

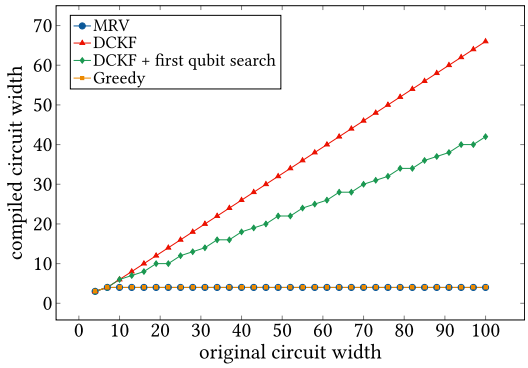

Fig. 6. Compiled circuit width against the original circuit width of quantum ripple carry adders.

devices while maintaining sufficient structural complexity to make classical simulation intractable. An interesting observation from Fig. 5 shows that the cycles chosen by [32], [33], [34] happened to be the critical points of the circuit reducibility, i.e., the number of cycles that just make the circuit irreducible. This suggests that the irreducibility factor in our framework may serve as a valuable reference for evaluating the balance required when designing quantum circuits with repeated cycles, especially in the context of demonstrating quantum supremacy in the future.

Another noteworthy point is the quantum circuit with 53 qubits and 20 cycles executed on Sycamore [35] exhibits a reducibility factor of 0.02, indicating that the circuit width can be reduced to 52. This observation aligns with the historical fact that one qubit on the Sycamore chip is non-functional, thereby breaking the well-designed structure of the circuit and leaving the room for qubit reuse.

2) Algorithm Benchmarking: In this section, we compare the performance of several algorithms across a range of benchmark circuits. Our evaluation focuses on four algorithms for dynamic circuit compilation aimed at minimizing circuit width: the MRV heuristic algorithm, the greedy heuristic algorithm, and two variants of the greedy algorithms proposed in [26] (referred to as DCKF and DCKF + first qubit search hereafter). Since the source code for the DCKF algorithms is not publicly available, we implemented these algorithms based on our understanding of the original paper [26]. For each benchmark instance, we employ two separate runs of the MRV algorithm, where the roles of roots and terminals in the algorithm are exchanged and we select the better output. Additionally, we leverage the stochastic nature of our greedy heuristic algorithm by executing it multiple times per instance and returning the best result.

a) Quantum Ripple Carry Adders: Fig. 6 illustrates the compilation results of quantum ripple carry adder circuits conducted using different heuristic strategies. Both the MRV and greedy heuristics consistently identify the optimal solution. In contrast, the results obtained by the DCKF algorithms display linear scaling with the original circuit width. This deficiency likely arises from its specific implementation: the process of establishing measurement orders within the DCKF algorithms

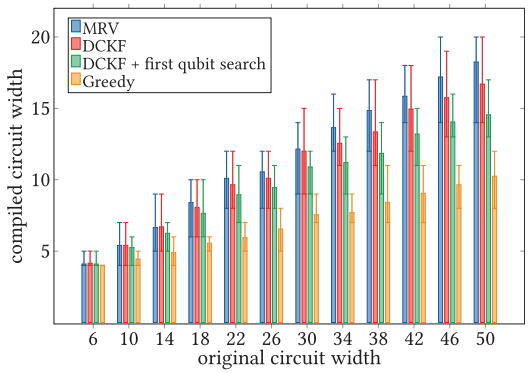

Fig. 7. Compiled circuit width against the original circuit width of the max-cut QAOA circuits with $p = 1$. The plotted error bars correspond to the maximum and minimum compiled width over 20 instances.

sometimes generates multiple local optima within a single iteration. Consequently, deterministic selections in these scenarios might lead to unfavorable measurement orders, significantly affecting its performance. This justifies the reasoning behind our inclusion of randomness within our greedy heuristic algorithm.

b) QAOA circuits for max-cut problem: Quantum approximate optimization algorithm (QAOA) [27] is a quantum algorithm designed to approximately solve classical combinatorial optimization problems. The QAOA unitary is composed of the alternative application of a mixing unitary and a problem unitary for p layers. Here, we assess the performance of different algorithms applied to QAOA circuits for solving the max-cut problem on random unweighted three-regular (U3R) graphs with $p = 1$. For each instance, we ran our greedy heuristic algorithm 10 times and recorded the best result. We evaluated four algorithms for each fixed qubit number on 20 random U3R graphs generated using the *NetworkX* package [36]. The results are presented in Fig. 7. It is evident that the average compiled width achieved by our greedy algorithm is consistently lower than that obtained using the DCKF algorithms for all qubit numbers. Moreover, as the number of qubits increases, this advantage becomes increasingly pronounced.

c) Random circuits: In addition to the previously studied structured circuits, we conducted numerical experiments with

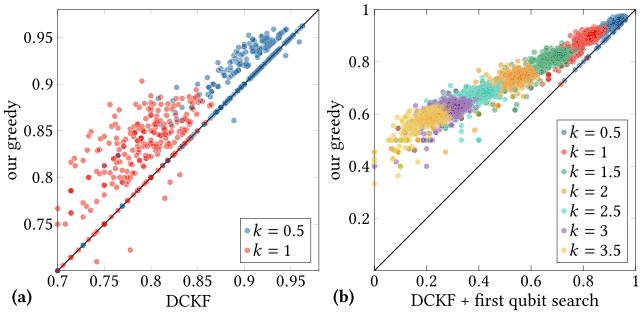

Fig. 8. (a) The reducibility factor of the randomly quantum circuits evaluated using our greedy heuristic algorithm and the DCKF algorithm. (b) The reducibility factor of the randomly IQP circuits evaluated using our greedy heuristic algorithm and the improved DCKF algorithm. The black diagonal line indicates the point at which the reducibility factors produced by both algorithms are equal.

random quantum circuits to evaluate algorithm performance over a wider range of scenarios. In these experiments, we fixed the ratio k between the number of two-qubit gates and the number of qubits. We then randomly selected a qubit number from [10, 80] and sampled the desired number of two-qubit gates to construct each circuit. We evaluated the reducibility factor using both our greedy heuristic and the DCKF algorithms on these random circuits. For each fixed ratio k, we sampled 300 random circuits and ran our greedy algorithm 15 times for each instance. The results in Fig. 8(a) demonstrate that our greedy heuristic outperforms the DCKF algorithm in approximately 98.5% of instances.

d) Random IQP circuits: To underscore the significance of handling circuits with commutable structures, we evaluated the reducibility factor of random IQP circuits using our greedy and the improved DCKF algorithms. We sampled 300 random IQP circuits for each fixed ratio and ran our greedy algorithm 10 times per instance. As depicted in Fig. 8(b), our greedy algorithm outperforms in nearly 100% of instances with an absolute advantage in 98.4% of cases.

3) Noisy Simulation: Error variability poses a challenge in near-term quantum hardware. By maximizing qubit reuse, we can consistently select qubits with better performance, thereby enhancing the circuit performance. To further demonstrate the practical efficacy of the proposed methods, we design a noisy simulation of an 11-qubit Bernstein-Vazirani (BV) algorithm, specifically targeting the real-world 11-qubit trapped-ion quantum computer reported in [37]. The secret bitstring of the BV algorithm is set to an all-one string.

In our simulation, we gradually reduce the number of qubits from 11 to 2 and map the logical qubits onto the physical qubits of the hardware, systematically eliminating a physical qubit with a higher error rate at each step. For dynamic circuit compilation, we always reuse the logical qubit q_0 for other qubits. The logical–physical qubit mapping and the probability of obtaining the correct outcome in each circuit are listed in Table II. The utilization of dynamic quantum circuit compilation enables us to reduce qubit usage by up to 82% while also improving the probability of achieving accurate results by 8%.

TABLE II
THE MAPPING OF LOGICAL QUBITS (q) TO PHYSICAL QUBITS (Q) AND THE PROBABILITY OF OBTAINING THE CORRECT OUTCOME IN NOISY SIMULATION

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}	prob.
BV_11	Q_3	Q_0	Q_2	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_1	66.2%
BV_10	Q_3	Q_0	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_1	-	66.7%
BV_9	Q_3	Q_0	Q_4	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_1	-	-	68.5%
BV_8	Q_3	Q_0	Q_4	Q_6	Q_7	Q_8	Q_9	Q_1	-	-	-	69.1%
BV_7	Q_3	Q_0	Q_4	Q_7	Q_8	Q_9	Q_1	-	-	-	-	70.5%
BV_6	Q_3	Q_0	Q_4	Q_7	Q_9	Q_1	-	-	-	-	-	71.5%
BV_5	Q_3	Q_0	Q_7	Q_9	Q_1	-	-	-	-	-	-	73.1%
BV_4	Q_3	Q_0	Q_7	Q_1	-	-	-	-	-	-	-	73.6%
BV_3	Q_3	Q_0	Q_1	-	-	-	-	-	-	-	-	74.1%
BV_2	Q_3	Q_1	-	-	-	-	-	-	-	-	-	74.3%

The simulations are performed using the "DensityMatrix" backend within our software toolkit. To incorporate realistic imperfections, we employ a simple depolarizing noise model parameterized by the experimentally reported gate fidelities of the trapped-ion device in [37]. After each reset or before each measurement, a depolarizing channel is applied with probability given by the reported state preparation and measurement (SPAM) infidelity of that physical qubit. Similarly, every single-qubit gate is followed by a depolarizing channel determined by that qubit's single-qubit gate fidelity. For two-qubit operations, we approximate the noise by applying independent depolarizing channels to each participating qubit, with error probabilities derived from the reported two-qubit fidelities. We further assume that CNOT gates admit the same performance as the native X-X gates.

Note that this example is intended as an illustrative study under certain simplifying assumptions. In practice, hardware-native gates in trapped-ion devices (e.g., Mølmer–Sørensen interactions [38]) are subject to a wide range of error mechanisms—including coherent over- or under-rotations, crosstalk, and correlated two-qubit errors [39], [40], [41]—that cannot be accurately captured by a simple depolarizing channel. In addition, mid-circuit measurement and reset operations exhibit distinct noise characteristics from standard SPAM errors and may introduce correlated errors on spectator qubits [42], [43], [44], making their accurate modeling highly hardware dependent. Nonetheless, for the purpose of this study, the simplified model provides a transparent and hardware-informed illustration of how dynamic circuit compilation can enhance performance under realistic error variability. Incorporating detailed hardware-specific noise models and exploring how dynamic compilation can be further optimized to mitigate their effects are left for future research.

IX. Related Work

Unqomp [45] is a procedure designed to automatically synthesize uncomputation within a given quantum circuit. It can also facilitate qubits reuse by repurposing the wire holding a correctly uncomputed ancilla to hold another ancilla. However, it is worth noting that the reclamation through uncomputation is only limited to ancillary qubits that do not require measurement. In our context, the reused qubits are not limited to ancillary

qubits, and we still need to retain the measurement outcomes of the qubits before reusing them.

DeCross et al. [26] investigated qubit-reuse compilation by leveraging the causal structure of the quantum circuits. They formulated the task of minimizing the required number of qubits as a constraint programming and satisfiability (CP-SAT) model. This model incorporates several constraints and is primarily utilized to numerically benchmark their heuristic algorithms on small-scale problems. In contrast, our approach utilizes a graph manipulation framework that induces direct binary integer programming formulation for qubit-reuse-optimal compilation. This framework not only enables formal proofs of optimality for several circuits but also forms the basis of our custom solver, which can integrate different heuristic strategies. Additionally, they proposed a greedy heuristic algorithm for approximate compilation. Our comparative analysis in Section VIII highlights the superior performance of our methods over their greedy heuristic across both structured and random quantum circuits. Notably, our framework addresses an open challenge emphasized in their work: the compilation of quantum circuits with commutable structures, which is an important feature in many modern quantum applications.

Hua et al. [24] explored the tradeoff between qubit reuse, fidelity, gate count, and circuit duration. They established two conditions for qubit reuse and designed two versions of compiler-assisted tools: one prioritizing qubit-saving and the other emphasizing SWAP reduction and fidelity improvement. Their experiments on superconducting quantum devices demonstrated notable improvements in qubit usage and circuit fidelity for specific applications. Our approach, on the other hand, centers on minimizing the required number of qubits—a scenario well-motivated by trapped ion quantum systems. Additionally, we contribute a more adaptable framework capable of flexible extensions to accommodate various optimization objectives. For instance, by fine-tuning the cost function within our greedy algorithm's scoring process, we can explore tradeoffs between circuit width, depth, and other related factors, essentially encompassing their qubit-saving approach. Furthermore, the optimal compilation strategies identified in our work can also serve as benchmarks for other variants of qubit-reuse compilation methods.

The work [25] introduced a formal SAT-based model for qubit reuse optimization on near-term quantum devices by combining a qubit reset–reuse model with an existing swap-insertion model. While their method provides provably optimal solutions with respect to both qubit count and swap-gate insertion, it relies on an off-the-shelf Z3-SMT solver and is computationally intensive, leading to scalability challenges as the number of qubits grows. In contrast, our framework is graph-based, focused on minimizing the number of qubits, and supports both theoretical optimality proofs and heuristic solver design.

X. Conclusion and Future Work

We conducted a systematic investigation into the dynamic circuit compilation, introducing the first framework for this task through graph manipulation. Our framework primarily targets qubit savings but is general enough to be adapted to other scenarios. The effectiveness of our approach was demonstrated through theoretical analyses of qubit-reuse–optimal compilations for various renowned quantum circuits, as well as numerical evaluation of our heuristic algorithms on a wide range of benchmark circuits. The dynamic circuit compilation explored in this work offers a complementary strategy to other circuit optimization techniques and can be seamlessly integrated with existing methods. In general, dynamic quantum circuit compilation has the potential to facilitate the implementation of quantum algorithms that exceeds the capacity of available devices, regardless of whether logical or physical qubits are involved. Our effort serves as timely contributions towards the practical implementation of large-scale quantum algorithms on quantum computers with limited resources.

Future work includes integrating our framework with hardware-specific considerations, such as native gate sets, connectivity constraints, and realistic noise models. These extensions, along with experimental demonstrations, will further enhance the practical relevance of dynamic circuit compilation and its integration into standard circuit compilation pipelines.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc., 35th Annu. Symp. Found. Comput. Sci.*, Piscataway, NJ, USA: IEEE Press, 1994, pp. 124–134.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. Twenty-Eighth Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA: Association for Computing Machinery, Jul. 1996, pp. 212–219.

[3] B. P. Lanyon et al., "Towards quantum chemistry on a quantum computer," *Nat. Chem.*, vol. 2, no. 2, pp. 106–111, Jan. 2010.

[4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017.

[5] S. Bravyi, R. Shaydulin, S. Hu, and D. Maslov, "Clifford circuit optimization with templates and symbolic Pauli gates," *Quantum*, vol. 5, Nov. 2021, Art. no, 580.

[6] M. Xu et al., "Quartz: Superoptimization of quantum circuits," in *Proc. 43rd ACM SIGPLAN Int. Conf. Program. Lang. Des. Implementation, (PLDI)*, New York, NY, USA: ACM, Jun. 2022, pp. 625–640.

[7] A. Xu, A. Molavi, L. Pick, S. Tannu, and A. Albarghouthi, "Synthesizing quantum-circuit optimizers," in *Proc. ACM Program. Lang.*, vol. 7. New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 835–859.

[8] E. Muñoz-Coreas and H. Thapliyal, "Quantum circuit design of a T-count optimized integer multiplier," *IEEE Trans. Comput.*, vol. 68, no. 5, pp. 729–739, May 2019.

[9] M. Zhang, T. Shi, W. Wu, and H. Sui, "Optimized quantum circuit of AES with interlacing-uncompute structure," *IEEE Trans. Comput.*, vol. 73, no. 11, pp. 2563–2575, Nov. 2024.

[10] P. Zhu, W. Ding, L. Wei, X. Cheng, Z. Guan, and S. Feng, "A variation-aware quantum circuit mapping approach based on multi-agent cooperation," *IEEE Trans. Comput.*, vol. 72, no. 8, pp. 2237–2249, Aug. 2023.

[11] D. Yu and K. Fang, "Symmetry-based quantum circuit mapping," *Phys. Rev. Appl.*, vol. 22, no. 2, 2024, Art. no. 024029.

[12] S. Li, X. Zhou, and Y. Feng, "Qubit mapping based on subgraph isomorphism and filtered depth-limited search," *IEEE Trans. Comput.*, vol. 70, no. 11, pp. 1777–1788, Nov. 2021.

[13] P. Zhu, S. Feng, and Z. Guan, "An iterated local search methodology for the qubit mapping problem," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2587–2597, Aug. 2022.

[14] K. Ye et al., "A mutual-influence-aware heuristic method for quantum circuit mapping," *IEEE Trans. Comput.*, vol. 73, no. 12, pp. 2855–2867, Dec. 2024.
[15] C. Zhang, A. B. Hayes, L. Qiu, Y. Jin, Y. Chen, and E. Z. Zhang, "Time-optimal qubit mapping," in *Proc. 26th ACM Int. Conf. Archit. Support Program. Lang. Operating Syst., ser. (ASPLOS)*, New York, NY, USA: ACM, 2021, pp. 360–374.
[16] A. Kole, S. Hillmich, K. Datta, R. Wille, and I. Sengupta, "Improved mapping of quantum circuits to IBM QX architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2375–2383, Oct. 2020.
[17] A. D. Córcoles et al., "Exploiting dynamic quantum circuits in a quantum algorithm with superconducting qubits," *Phys. Rev. Lett.*, vol. 127, no. 10, Aug. 2021, Art. no. 100501.
[18] J. M. Pino et al., "Demonstration of the trapped-ion quantum CCD computer architecture," *Nature*, vol. 592, no. 7853, pp. 209–213, Apr. 2021.
[19] G. Q. Ai, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, no. 7949, pp. 676–681, Feb. 2023.
[20] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Phys. Rev. A*, vol. 86, no. 3, Sep. 2012, Art. no. 032324.
[21] Z. Ni et al., "Beating the break-even point with a discrete-variable-encoded logical qubit," *Nature*, vol. 616, no. 7955, pp. 56–60, 2023.
[22] R. Govindarajan, H. Yang, J. Amaral, C. Zhang, and G. Gao, "Minimum register instruction sequencing to reduce register spills in out-of-order issue superscalar architectures," *IEEE Trans. Comput.*, vol. 52, no. 1, pp. 4–20, Jan. 2003.
[23] A. Paler, R. Wille, and S. J. Devitt, "Wire recycling for quantum circuit optimization," *Phys. Rev. A*, vol. 94, no. 4, Oct. 2016, Art. no. 042337.
[24] F. Hua et al., "CaQR: A compiler-assisted approach for qubit reuse through dynamic circuit," in *Proc. 28th ACM Int. Conf. Archit. Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA: Association for Computing Machinery, 2023, pp. 59–71
[25] S. Brandhofer, I. Polian, and K. Krsulich, "Optimal qubit reuse for near-term quantum computers," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, Bellevue, WA, USA, 2023, pp. 859–886.
[26] M. DeCross, E. Chertkov, M. Kohagen, and M. Foss-Feig, "Qubit-reuse compilation with mid-circuit measurement and reset," *Phys. Rev. X*, vol. 13, Dec. 2023, Art. no. 041057.
[27] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014, *arXiv:1411.4028*.
[28] H. J. Briegel, D. E. Browne, W. Dür, R. Raussendorf, and M. V. den Nest, "Measurement-based quantum computation," *Nat. Phys.*, vol. 5, no. 1, pp. 19–26, Jan. 2009.
[29] D. Shepherd and M. J. Bremner, "Temporally unstructured quantum computation," *Proc. Roy. Soc. A: Math., Physical Eng. Sci.*, vol. 465, no. 2105, pp. 1413–1439, Feb. 2009.
[30] Y. Li, Y. Qiao, A. Wigderson, Y. Wigderson, and C. Zhang, "Connections between graphs and matrix spaces," 2022.
[31] L. Gurobi Optimization, "Gurobi optimizer reference manual," [Online]. Available: http://www.gurobi.com
[32] A. Morvan et al., "Phase transition in random circuit sampling," 2023.
[33] Y. Wu et al., "Strong quantum computational advantage using a superconducting quantum processor," *Phys. Rev. Lett.*, vol. 127, no. 18, Oct. 2021, Art. no. 180501.
[34] Q. Zhu et al., "Quantum computational advantage via 60-qubit 24-cycle random circuit sampling," *Sci. Bull.*, vol. 67, no. 3, pp. 240–245, Feb. 2022.
[35] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019.
[36] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proc. 7th Python Sci. Conf. (Scipy)*, Pasadena, CA, USA, Aug. 2008, pp. 11–15.
[37] K. Wright et al., "Benchmarking an 11-qubit quantum computer," *Nat. Commun.*, vol. 10, no. 1, Nov. 2019, Art. no. 5464.
[38] K. Mølmer and A. Sørensen, "Multiparticle entanglement of hot trapped ions," *Phys. Rev. Lett.*, vol. 82, no. 9, pp. 1835–1838, Mar. 1999.
[39] C. J. Ballance, T. P. Harty, N. M. Linke, M. A. Sepiol, and D. M. Lucas, "High-fidelity quantum logic gates using trapped-ion hyperfine qubits," *Phys. Rev. Lett.*, vol. 117, no. 6, Aug. 2016, Art. no. 060504.
[40] J. P. Gaebler et al., "High-fidelity universal gate set for be 9 + ion qubits," *Phys. Rev. Lett.*, vol. 117, no. 6, Aug. 2016, Art. no. 060505.
[41] C. Fang, Y. Wang, S. Huang, K. R. Brown, and J. Kim, "Crosstalk suppression in individually addressed two-qubit gates in a trapped-ion quantum computer," *Phys. Rev. Lett.*, vol. 129, no. 24, Dec. 2022, Art. no. 240504.
[42] L. C. G. Govia, P. Jurcevic, C. J. Wood, N. Kanazawa, S. T. Merkel, and D. C. McKay, "A randomized benchmarking suite for mid-circuit measurements," *New J. Phys.*, vol. 25, no. 12, Dec. 2023, Art. no. 123016.
[43] D. Hothem, J. Hines, C. Baldwin, D. Gresh, R. Blume-Kohout, and T. Proctor, "Measuring error rates of mid-circuit measurements," *Nat. Commun.*, vol. 16, no. 1, Jul. 2025, Art. no. 5761.
[44] J. Hines and T. Proctor, "Pauli noise learning for mid-circuit measurements," *Phys. Rev. Lett.*, vol. 134, no. 2, Jan. 2025, Art. no. 020602.
[45] A. Paradis, B. Bichsel, S. Steffen, and M. Vechev, "Unqomp: Synthesizing uncomputation in quantum circuits," in *Proc. 42nd ACM SIGPLAN Int. Conf. Program. Lang. Des. Implementation, (PLDI)*, New York, NY, USA: ACM, 2021, pp. 222–236.

Kun Fang (Member, IEEE) received the B.S. degree in mathematics from Wuhan University, in 2015, and the Ph.D. degree in quantum information from the University of Technology Sydney, in 2018. After that, he worked as a Postdoctoral Researcher with the University of Cambridge and the University of Waterloo, from 2018 to 2020. He served as a Senior Researcher and a Tech Lead with the Institute for Quantum Computing, Baidu, from 2020 to 2023. He is currently an Assistant Professor with the School of Data Science, The Chinese University of Hong Kong, Shenzhen. His research interests include understanding the capabilities and limitations of quantum resources, as well as their usage in quantum computation and quantum communication.

Munan Zhang received the B.S. degree in physics from Nankai University, in 2021, and the M.S. degree in physics from the National University of Singapore, in 2022. He is currently working toward the Ph.D. degree with the School of Data Science, The Chinese University of Hong Kong, Shenzhen. His research interests include quantum information and quantum computation.

Ruqi Shi received the B.S. degree in applied physics from Nankai University, in 2021, and the M.S. degree in photonic engineering from Ghent University, in 2022. She is currently working toward the Ph.D. degree with Photonics Research Group, Ghent University-IMEC. Her research interest includes photonic Ising machine.

Yinan Li received the B.S. degree in mathematics from Wuhan University, in 2014, and the Ph.D. degree from the University of Technology Sydney, in 2018. He was a Postdoctoral Researcher with the Centrum Wiskunde & Informatica, the National Research Institute for Mathematics and Computer Science, The Netherlands, and also affiliated with the Research Center for Quantum Software (QuSoft). From 2020 to 2022, he was a Designated Assistant Professor with the Graduate School of Mathematics, Nagoya University. He is currently an Assistant Professor with the School of Artificial Intelligence, Wuhan University. He research interests include mathematical foundation of quantum computation, particularly, quantum (and classical) algorithms and quantum information theory.