Quantum Information & Computation
ISSN 1533-7146 Volume 25 (2025) pp. 216–231

Cerebration Science Publishing

Article

Width Optimization of Quantum Circuit Based on Reuse-Aimed Quantum Circuit Transformation

Haotian Tang [1], Fei Ding [1], Xueyun Cheng [1], Shuxian Zhao [2] and Zhijin Guan [1,3,*]

[1] School of Artificial Intelligence and Computer Science, Nantong University, Nantong 226019, China; 2330320036@stmail.ntu.edu.cn (H.T.); dingfei_ntu@ntu.edu.cn (F.D.); chen.xy@ntu.edu.cn (X.C.)
[2] Jiangsu Huihuan Environmental Protection Technology Co., LTD, Nantong 226010, China
[3] College of Engineering the Internet of Things, Taihu University, Wuxi 214063, China
* Correspondence author: guan.zj@ntu.edu.cn

Received date: 7 March 2025; Accepted date: 16 April 2025; Published online: 16 May 2025

Abstract: Due to constraints in current quantum hardware manufacturing, the number of available qubits remains limited. This limitation poses significant challenges to the feasibility of quantum computation. Qubit reuse, supported by the hardware-enabled dynamic quantum circuit, has emerged as a promising method for optimizing the width of quantum circuits. However, enabling width optimization through qubit reuse for circuits that lack qubit reuse opportunities remains a challenging problem. This paper aims to address this challenge by introducing RaQCT (Reuse-aimed Quantum Circuit Transformation), an algorithm for converting non-reusable qubit pairs into reusable ones. Moreover, this paper proposes a more general method for identifying reusable qubits based on the perspective of the execution sequence of the quantum gates. Based on the proposed identification method of reusable qubit pairs and RaQCT, this paper proposes width optimization strategies based on a tree structure. Compared with state-of-the-art qubit reuse methods, our proposed methods enable further circuit width optimization, which proves effective. Furthermore, the comprehensive set of methods proposed in this paper for handling qubit reuse does not require specific differentiation of circuit types. Meanwhile, it is applicable to both static and dynamic circuits, thereby also demonstrating generality.

Keywords: qubit reuse; mid-circuit measurement and reset; quantum circuit width optimization; quantum circuit transformation

1. Introduction

Quantum computing, as an advanced computational technology, integrates concepts from quantum physics, computer science, and information theory. Compared to classical computing, quantum computing has demonstrated significant advantages in fields such as artificial intelligence [1,2], materials chemistry [3,4], and optimization problems [5]. At present, quantum computing devices remain in the NISQ (Noisy Intermediate-Scale Quantum) era [6]. However, due to the constraints of hardware manufacturing [7], current quantum computers face strict limitations on the number of available qubits. As quantum computing technology advances, the number of qubits required for designing complex quantum circuits gradually increases, but existing devices struggle to meet this demand. To fully harness the potential of quantum computing in practical applications, researchers are actively exploring more efficient methods of qubit utilization [8,9].

Most quantum algorithms are traditionally represented as static quantum circuits [10], where a series of quantum gates perform specific operations on qubits, with the computational results ultimately extracted through measurement at the end of the circuit. Recently, numerous quantum hardware manufacturers have introduced devices supporting mid-circuit measurement and reset [11–13], including qubit architectures based on superconducting circuits and ion traps [14,15]. With this support, qubits can be measured and reset during circuit execution, allowing further circuit evolution based on measurement outcomes. This type of circuit, which allows mid-circuit measurement and reset, is also referred to as dynamic quantum circuit [16,17]. Initially, the primary application of mid-circuit measurement and reset was quantum error correction [18], but it has also introduced a novel circuit optimization method known

DOI: 10.2478/qic-2025-0011

as qubit reuse [19]. Through mid-circuit measurement and reset, qubits that have completed their operations can be measured and reset, and replace qubits to which no operation has been applied. Thus, qubit reuse can reduce the width of a quantum circuit. The quantum circuit width refers to the number of qubits used within a quantum circuit [20]. However, after a circuit is mapped from a logical circuit to quantum hardware [21–23], the insertion of numerous SWAP gates can diminish the opportunities for qubit reuse [24]. Therefore, qubit reuse is typically applied to logical quantum circuits before mapping to quantum hardware. As shown in Figure 1, after g_0 is completed, q_0, the control qubit of g_0, has no further operations while q_1, the control qubit of another gate g_1, has not yet begun any operations; thus q_0 can be measured and reset to replace q_1.

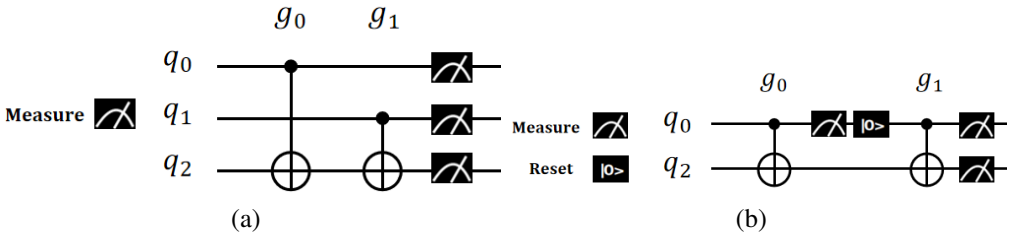


Figure 1. Reduce qubit of circuit by qubit reuse: (**a**) Original circuit, (**b**) Reuse q_0 to replace q_1.

The execution of some quantum algorithms currently requires a large number of qubits, which far exceed the capacity of current NISQ devices. Therefore, qubit reuse can be leveraged to optimize the width of these quantum algorithms. Qubit reuse was first proposed in [25], which illustrated that quantum computation can actually make full use of qubits. This work provided a foundation for subsequent research on qubit reuse. Then, as quantum computing hardware continued to evolve and develop, in [26], they proposed a qubit reuse strategy through mid-circuit measurement and reset, providing both an exact optimization model and a heuristic for obtaining the width optimization solutions of quantum circuits. Moreover, their work introduced the concept of dual circuits to further improve the effect of width optimization for quantum circuits. In [24], they designed QS-CaQR, a compiler-assisted tool for qubit reuse, which presented different schemes to trade off the quantum circuit width with other metrics. Furthermore, in [27], they proposed a SAT-based model to combine qubit reuse with quantum circuit mapping, which optimized different performance metrics like circuit depth and SWAP insertions. In [28], they presented a systematic study of dynamic quantum circuit compilation. Their work established the first general framework for optimizing the dynamic circuit compilation via graph manipulation, as well as addressing the circuits with commutable structures. In [29], they proposed two quantum circuit resizing algorithms. The first algorithm leveraged gate-dependency relations and the second found resizing opportunities through numerical instantiation and synthesis. In [30], they revisited the qubit reuse and introduced qubit dependency graphs as a key abstraction for width optimization. In [31], they proposed the GidNET algorithm which leverages the directed acyclic graph of a quantum circuit and its matrix representation to identify pathways for qubit reuse.

The above works provided inspiration for us to explore the problem of qubit reuse. The key aspect of addressing qubit reuse is ensuring that the gates applied to the qubit to be reused and the qubit to be replaced must satisfy a specific relationship if we view qubit reuse from the specific order in which operations are executed: quantum gate execution sequence. By taking the above factor into account, it becomes possible to identify qubits to be reused and qubits to be replaced, thereby optimizing circuit width. Actually, the inability to optimize the circuit width arises from the absence of reusable and substitutable qubits within the circuit. It is due to the execution sequence of quantum gates failing to meet specific conditions that the reusable and substitutable qubits cannot be found in the circuit. Therefore, by adjusting the execution sequence of quantum gates within the circuit while ensuring circuit equivalence, it becomes possible to transform non-reusable qubits into reusable ones. Specifically, the key to solving this problem is to determine which quantum gates need execution sequence adjustments and how to implement them.

This paper aims to address the issue of qubit reuse through a general methodology, including both the identification of qubit reuse opportunities and the strategy for width optimization. Following are our main contributions:

- We introduce a sequential partitioning method for quantum circuits. Then, a general method for identifying reusable qubits from the perspective of the execution sequence of the quantum gates is presented. This method is applicable to both static and dynamic circuits.

- In order to optimize circuit width better, a reuse-aimed quantum circuit transformation (RaQCT) algorithm is proposed. Through utilizing RaQCT, non-reusable qubit pairs can be converted into reusable ones at appropriate moments.
- For optimizing the width of quantum circuits, we design a tree-structure model for width optimization. We view the process of optimizing width as a nodes generation process of a tree.
- Based on the proposed tree-structure model, we provide an exact algorithm and a heuristic algorithm for width optimization.
- Our experiments show that we obtain further optimized width for circuits on average through our proposed RaQCT. Moreover, the experimental results indirectly demonstrate that our comprehensive approach exhibits greater generality, as it does not require differentiation among types of quantum circuits.

The organizational structure of this article is as follows: In Section 2, we investigate the general method for identifying reusable qubits, and the RaQCT for converting non-reusable qubits into reusable ones. In Section 3, we focus on studying the circuit width strategy method based on the tree structure. In Section 4, we experimentally validate the effectiveness of the proposed strategies in decreasing circuit width. Finally, we conclude the paper in Section 5.

2. Reuse-Aimed Quantum Circuit Transformation

In this section, we present a generalized method for the identification of reusable qubit pairs based on the execution sequence of the quantum gates. However, in some scenarios, quantum circuits may not be able to exploit qubit reuse opportunities using this general method. To overcome this limitation, we propose the RaQCT algorithm, designed to convert non-reusable qubit pairs into reusable ones.

2.1. Identification of Reusable Qubit Pairs

In a quantum circuit, assuming that Q is the set of qubits and G is the set of quantum gates, Q=$\{q_0,...,q_i,...,q_{n-1}\}$, G=$\{g_0,...,g_j,...,g_{n-1}\}$. In this paper, there are only single-qubit gates and two-qubit gates in quantum circuits.

Definition 1. *For two different qubits q_i and q_j in a quantum circuit, if q_i can be reused to replace q_j, then the qubit pair (q_i,q_j) is a reusable qubit pair.*

For a qubit pair (q_i,q_j), if q_i is to be reused for replacing q_j, in addition to ensuring that the gates applied to q_i have been completed, the execution sequence of the gates applied to q_j must also be taken into account. Thus, the identification of reusable qubit pairs in a circuit depends on the execution sequence information of quantum gates. Based on the considerations above, assigning each quantum gate independent execution sequence information would be highly beneficial for identifying reusable pairs within the circuit in an intuitive and generalizable way.

Definition 2. *In a quantum circuit with N quantum gates, the N quantum gates are sequentially divided into N time steps from left to right. The order of quantum gates that can be executed in parallel can be arranged arbitrarily. This partitioning method is called the sequential partitioning method for quantum circuits.*

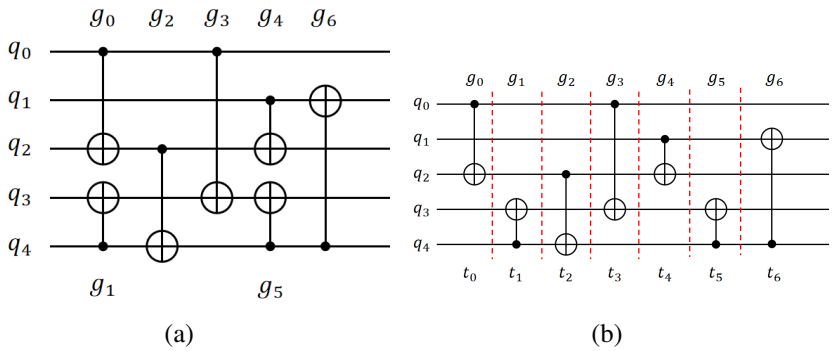
In Figure 2, we demonstrate an example of the sequential partitioning method. The partitioned circuit in Figure 2b is transformed from the original circuit in Figure 2a. After performing the sequential partitioning on the circuit in Figure 2a, we obtain an ordered execution sequence of gates (g_0, g_1, g_2, g_3, g_4, g_5, g_6). In this paper, we present a method for identifying qubits that can be reused and replaced from the perspective of the specific gate execution sequence. The method we present offers greater generality by focusing on the circuit itself. Furthermore, the method we propose is applicable to both static and dynamic circuits.

Theorem 1. *In a sequentially partitioned quantum circuit, the necessary and sufficient condition for reusing q_i to replace q_j is that the execution sequence of all gates applied to q_i must precede the execution sequence of all gates applied to q_j.*

Proof. (Necessity) Assuming that within the execution sequence of gates applied to q_i, there exist gates whose execution sequence is later than the execution sequence of gates applied to q_j. Since q_i needs to be measured and reset

before reuse, when to reuse q_i for replacing q_j, the gates applied to q_i, whose execution sequence follows that of gates applied to q_j, are disrupted due to the mid-circuit measurement and reset. At this point, the circuit is no longer equivalent to the original circuit. Therefore, the execution sequence of all gates applied to q_i must precede the execution sequence of all gates applied to q_j.

(Sufficiency)If all gates applied to q_i are completed, then q_i is no longer used, thus it is feasible to measure and reset q_i for reusing it to replace q_j. □



(a) (b)

Figure 2. Sequential partitioning for circuit: (**a**) Original circuit, (**b**) Circuit after sequential partitioning.

For any qubit pair in the circuit, as long as it satisfies the above condition, it is considered a reusable pair. Thus, it is obvious that there should not be any gate between q_i and q_j. In Figure 2b, there is no gate between q_0 and q_1, and the execution sequence of g_0 and g_3 is before the execution sequence of g_4 and g_6. Accordingly, q_0 can be reused to replace q_1. Therefore, we only need to insert a set of mid-circuit measurement and reset operation after the last operation, g_3, on q_0 and then apply the operations originally acting on q_1 to q_0 after the mid-circuit measurement and reset. This transforms the original circuit into an equivalent circuit with fewer qubits. The equivalent circuit with fewer qubits is shown in Figure 3. We emphasize that the above-introduced method for identifying reusable qubits is applicable to both static and dynamic circuits. This is because, regardless of whether the qubit pairs are in a static or dynamic circuit, they must follow the essence of qubit reuse, which is the temporal ordering of quantum gate executions. As long as the qubit pairs satisfy the conditions we describe, they can form a reusable pair. Thus, our identification method for reusable qubit pairs is more general.

It is worth noting that the depth of the original circuit in Figure 2a is 4, and the depth of the equivalent circuit in Figure 3 is 6 (including the mid-circuit measurement and reset). Since the introduced mid-circuit measurement and reset operations inevitably increase circuit depth, qubit reuse is actually a circuit optimization method that trades circuit depth for a reduction in the number of qubits.

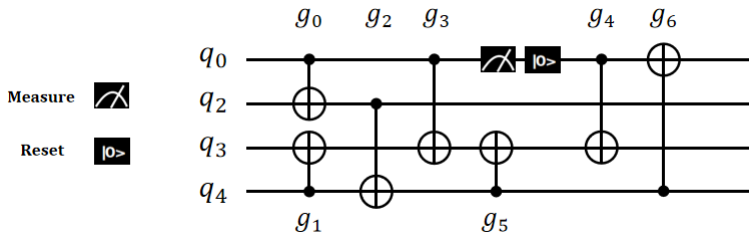


Figure 3. Equivalent circuit after reuse q_0 for replacing q_1.

2.2. Qubit Reuse Based on RaQCT

2.2.1. Problem Analysis

In the quantum circuit shown in Figure 2b, for the qubit pair (q_0,q_4), the gates applied to q_0 are g_0 and g_3, while the gates applied to q_4 are g_1, g_2, g_5 and g_6. According to the identification method for reusable qubit pairs in Section 2.1, (q_0,q_4) is not a reusable pair, and q_0 cannot be reused to replace q_4. However, according to the exchange rules of the quantum gate execution sequence provided in [32,33], two quantum gates with adjacent execution sequences can exchange their execution sequence without affecting the equivalence of the quantum circuit in certain scenarios. For example, if two adjacent CNOT gates in the execution sequence share the same control or target qubit, they can

exchange their execution sequence. Moreover, if the two adjacent CNOT gates act on completely different qubits, they can also exchange their execution sequence.

According to the aforementioned exchange rules, g_3 can exchange its execution sequence sequentially with g_2 and g_1 while preserving circuit equivalence. The transformed circuit is shown in Figure 4. In the transformed circuit, the execution sequence of g_0 and g_3 is before the execution sequence of g_1, g_2, g_5, g_6. The condition for reusing q_0 to replace q_4 is satisfied. Therefore, we only need to insert a set of mid-circuit measurement and reset after the last operation, g_3, on q_0 and then apply the operations originally acting on q_4 to q_0 after the mid-circuit measurement and reset. The equivalent circuit in which q_0 is reused to replace q_4 is shown in Figure 5.

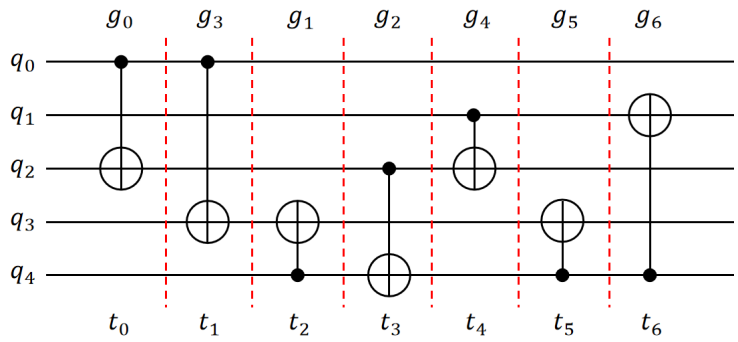


Figure 4. Circuit after transformation.

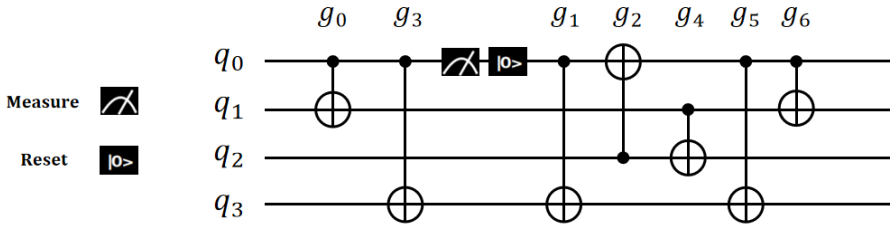


Figure 5. Equivalent circuit after reuse q_0 for replacing q_4.

For the qubit pair (q_i,q_j), for which q_i cannot be reused to replace q_j, it is possible to adjust the execution sequence of gates applied to q_i and q_j in the quantum circuit so that the execution sequence of gates applied to q_i and q_j can satisfy the condition for reusing q_i to replace q_j. Due to the complex sequential relationships between the gates applied to q_i and q_j, obtaining a gate execution sequence adjustment scheme that satisfies the reuse condition for q_i is relatively difficult. Hence, accurately identifying the quantum gates that require execution sequence adjustments is of importance. This point is especially critical for circuits in which reusable pairs cannot be obtained through the reusable pair identification method.

Definition 3. *In a sequentially partitioned quantum circuit, for the pair (q_i,q_j) where q_i cannot be reused to replace q_j, the gates applied to q_i whose execution sequence precedes all gates applied to q_j are called preceding gates. The gates applied to q_i whose execution sequence comes after any gate applied to q_j are called succeeding gates.*

Taking the pair (q_0,q_4) in the circuit of Figure 3 as an example, we mark the gates acting on q_0 with red dashed lines and those acting on q_4 with green dashed lines. The circuit being marked is shown in Figure 6. According to Definition 3, the execution sequence of g_0 precedes the execution sequence of all gates applied to q_4, thus g_0 is a preceding gate. The execution sequence of g_3 is after g_1 and g_2, therefore, g_3 is a succeeding gate. Obviously, according to the identification method for reusable qubit pairs, q_0 cannot be reused to replace q_4.

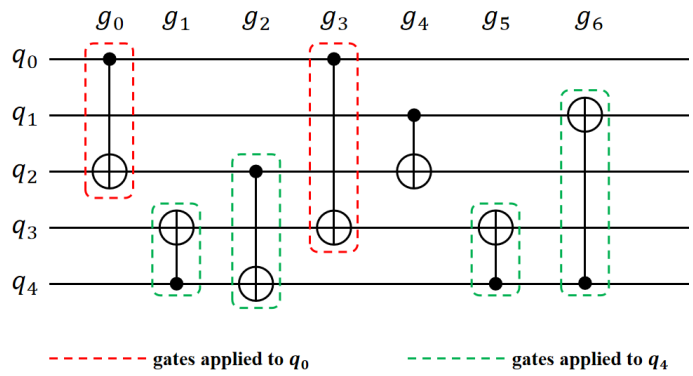


Figure 6. Preceding gates and succeeding gates of qubit pair (q_0,q_4) in circuit of Figure 3.

Since the execution sequence of the succeeding gates does not satisfy the reuse condition for q_i, all succeeding gates must be converted into preceding gates. During the process of converting succeeding gates into preceding gates, equivalent transformations need to be applied to the circuit. Therefore, the process of converting succeeding gates into preceding gates is essentially the process of equivalence transformation of the quantum circuit, where enabling q_i to satisfy the reuse condition is actually an equivalent transformation of the quantum circuit to make q_i reusable.

2.2.2. RaQCT Algorithm

From the analysis in Section 2.2.1, in order to enable q_i to be reused to replace q_j, we can apply an equivalent transformation to the circuit. Since the execution sequence of the succeeding gates does not satisfy the reuse condition for q_i, an equivalent transformation needs to be applied to the circuit to convert the succeeding gates into preceding gates. We propose the following two rules for converting succeeding gates into preceding gates.

Rule 1 For a pair (q_i,q_j), if the execution sequence of a succeeding gate is able to precede the execution sequence of all gates applied to q_j through circuit equivalent transformations, then this succeeding gate can be converted into a preceding gate.

Rule 2 For a pair (q_i,q_j), if the execution sequence of a succeeding gate is unable to precede the execution sequence of all gates applied to q_j through circuit equivalent transformations, then this succeeding gate cannot be converted into a preceding gate, and the reuse condition for q_i cannot be satisfied.

The following are the specific steps for converting a succeeding gate into a preceding gate.

Step 1: Identify the gate applied to q_j that has an execution sequence preceding the current succeeding gate and is closest to it.

Step 2: Move the execution sequence of the current succeeding gate forward. If this makes the execution sequence of the current succeeding gate precede the gate applied to q_j identified in the first step, check whether the current succeeding gate has been successfully converted into a preceding gate. If the current succeeding gate has already been converted into a preceding gate, the circuit transformation ends. If not, return to Step 1. If the execution sequence of this succeeding gate is unable to move forward, turn to Step 3.

Step 3: Move the execution sequence of the gate applied to q_j identified in the first step backward. If this makes the execution sequence of the succeeding gate precede the execution sequence of the gate applied to q_j identified in the first step, check whether the succeeding gate has been successfully converted into a preceding gate. If this succeeding gate has been converted into a preceding gate, the circuit transformation ends. If not, return to Step 1. If it is impossible to make the execution sequence of the succeeding gate precede the execution sequence of the gate applied to q_j by moving the execution sequence of the gate applied to q_j backward, the circuit transformation ends, and q_i does not satisfy the reuse condition.

In Step 1, the circuit is scanned from the current succeeding gate toward earlier positions in the execution sequence to locate the nearest gate applied to q_j. Given a circuit with n gates, this scan may, in the worst case, traverse the entire circuit from end to beginning. Hence, the worst-case time complexity of Step 1 is $O(n)$. In Step 2 and Step 3, we assume that each gate movement (regardless of whether it is forward or backward) requires constant time, i.e., $O(1)$. In the worst case, the current succeeding gate may need to be moved from the end of the circuit to the beginning, resulting in $O(n)$ gate movements. In the worst case, the process of converting a succeeding gate into a preceding gate may require up to $O(n)$ iterations (each involving repeated execution of Step 1 , Step 2 and Step 3) before successfully

completing or concluding that the condition cannot be met. Therefore, the worst-case time complexity for converting a succeeding gate into a preceding gate is $O(n^2)$. In practice, the worst-case time overhead is rarely encountered.

For a pair (q_i,q_j), in order to make q_i satisfy the reuse condition, the above strategy can be applied sequentially to all succeeding gates in the circuit until all succeeding gates are converted into preceding gates. During the process of RaQCT performing equivalence transformations on the circuit, various exchange rules [32,33] for execution sequence of gates are applied. These exchange rules are applied to quantum gates executed in series or parallel, depending on the specific situation. It should be particularly noted that a set of mid-circuit measurement and reset is considered as a complete operation, and no gate can be exchanged in execution sequence with this set of operations. Based on the identification method for reusable qubit pairs and the above proposed strategy, this paper presents a quantum circuit transformation algorithm to enable a qubit to be reused. In this algorithm, during any equivalent transformation of the circuit, if it is impossible to convert a succeeding gate into a preceding gate, the algorithm terminates, and q_i cannot be reused to replace q_j. If after several equivalent transformations of the circuit, the execution sequence of the gates applied to q_i and q_j satisfies the reuse condition for q_i, the algorithm terminates, and q_i can be reused for replacing q_j. This algorithm ultimately outputs a flag indicating whether q_i is reusable. The following is the pseudocode for the Algorithm 1.

Algorithm 1: Reuse-aimed Quantum Circuit Transformation

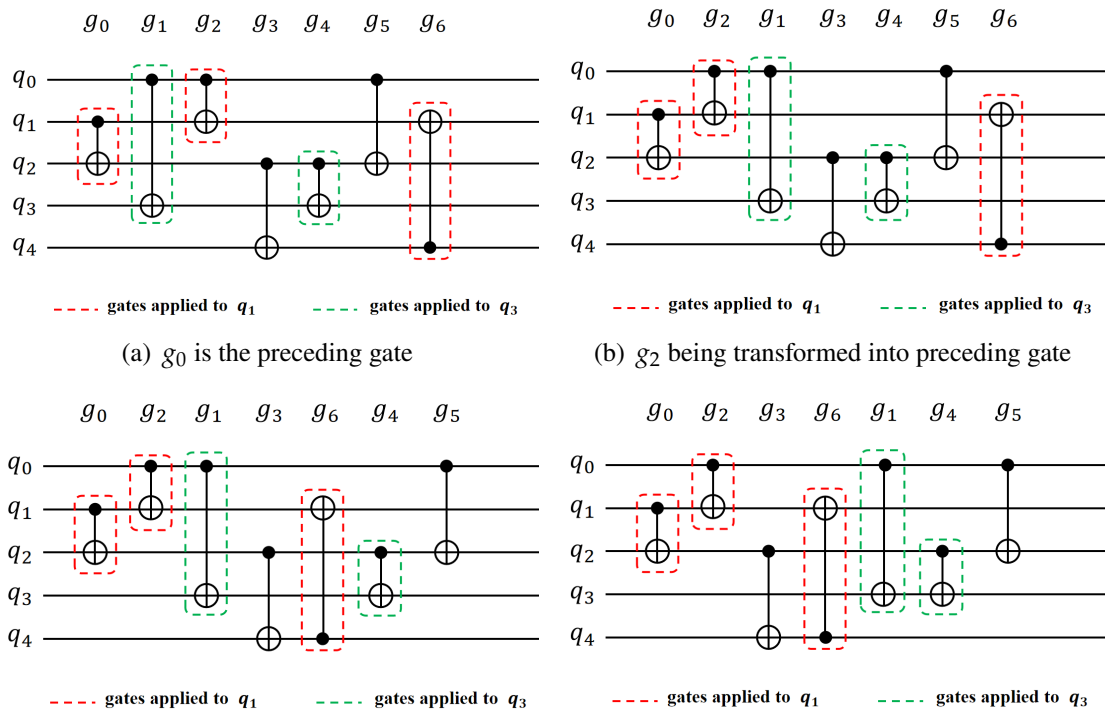
Input: qubit pair(q_i,q_j), quantum circuit list Cir =[g_0, g_1,...], list of gate applied to q_i L_{q_i}=[g_s,...]
Output: reuse flag of q_i $flag$

```
flag ← 1; // Initialize the reuse flag of q_i
foreach gate in L_{q_i} do
    while preceding_gate_judgment(gate) == 0 do
        if circuit_transformation(gate, Cir) == 1 then
            continue;// Judge if this gate can be converted into preceding gate
        end
        else
            flag ← 0; // Mark q_i as non-reusable
            break;
        end
    end
    if flag == 0 then
        break; // Check reuse flag
    end
    else
        continue; // Current succeeding gate can be converted into preceding gate,
            continue to judge the next gate in L_{q_i}
    end
end
return flag
```

2.2.3. Example

Based on the pseudocode provided in Section 2.2.2, we execute the qubit pair (q_1,q_3) in the circuit shown in Figure 7a, where q_1 cannot be reused to replace q_3. L_{q1} = [g_0, g_2, g_6].

First, initialize the flag as 1. In Figure 7a, the execution sequence of g_0 precedes the execution sequence of all gates applied to q_3, making g_0 a preceding gate. In Figure 7a, g_2 is a succeeding gate; it needs to be converted into a preceding gate. Since g_1 and g_2 share the same control qubit, g_2 can exchange its execution sequence with g_1. In Figure 7b, g_2 has already been converted into the preceding gate, and no further adjustment of the execution sequence of g_2 is needed. In Figure 7b, g_6 is a succeeding gate. Since the qubits operated on by g_6 are distinct from those operated on by g_5 and g_4, g_6 can sequentially exchange its execution sequence with g_5 and g_4, making the execution sequence of g_6 precede the execution sequence of g_4. As shown in Figure 7c, g_6 is still a succeeding gate. At this point, the forward movement of g_6 is blocked, but g_1 can sequentially exchange its execution sequence with g_3 and g_6 according to the exchange rules. In Figure 7d, all succeeding gates have been converted into preceding gates, and q_1 can be reused for replacing q_3.



(a) g_0 is the preceding gate

(b) g_2 being transformed into preceding gate

(c) g_6 still being succeeding gate

(d) All succeeding gates being transformed into preceding gates

Figure 7. Example of execution process of RaQCT.

3. Width Optimization Method Based on Tree Structure Model

Quantum circuits can achieve width optimization through qubit reuse. In this section, we introduce a tree-structure-based model for width optimization. Building upon the tree-structure model of width optimization, we propose both an exact algorithm and a heuristic algorithm for circuit width optimization, aiming to minimize the number of qubits required for circuit execution.

3.1. Tree-Structure Model for Width Optimization

For an original circuit C_n with width n, if there exists a reusable qubit pair (q_i,q_j), then the mid-circuit measurement and reset can be inserted after the last operation applied to q_i, allowing q_i to replace q_j. This updates the original circuit C_n to an equivalent circuit C_{n-1} with width $n-1$. As shown in Figure 8, if the original circuit with width n is viewed as the root node of a tree, and the equivalent circuit with width $n-1$ is viewed as a child node of the root, then the transformation from the original circuit to the width-optimized circuit can be considered as a node generation process in the tree. Therefore, if the original circuit C_n with width n has k reusable pairs p_i(for i = 0,1,...,k $-$ 1), then the original circuit C_n can be updated to k equivalent circuits C_{n-1} with width $n-1$ based on each reusable pair (since the width-optimized circuits are functionally equivalent, no further distinction is made here).

Based on the above analysis, the width optimization process of a quantum circuit can be viewed as the continuous generation of nodes in a tree. As shown in Figure 8b, by repeatedly generating nodes in this way, the process continues until no new nodes can be generated in the l-th layer of the tree. The width optimization process ends, and the width reaches the minimum value $n-l+1$ by using the identification method for reusable qubit pairs and the RaQCT in Section 2.2. However, as the scale of the circuit expands, the number of qubit pairs that need to be evaluated increases accordingly. If the width-optimized circuit for each layer is still obtained precisely in the aforementioned manner, the time required to achieve the optimal width will significantly increase. Therefore, exploring a method that can accelerate the width optimization process while ensuring the quality of the width-optimized solution (smaller width) is crucial.

If a quantum circuit contains a substantial number of reusable pairs, it implies that the interactions between qubits within the circuit are relatively less dense. Furthermore, since the identification of reusable pairs depends on the identification method and RaQCT, a higher number of reusable pairs in a circuit also reflects considerable adjustability in the execution sequence of quantum gates within that circuit. These are potential features that enable

continuous width optimization of a circuit. Therefore, as long as a circuit contains a sufficient number of reusable pairs, the next-generation circuits will potentially inherit the characteristic of having many reusable pairs. As shown in Figure 9, after generating all nodes in the current layer of the tree, the number of reusable pairs for all equivalent circuits in the current layer is calculated. The equivalent circuit with the most reusable pairs is selected to generate the next layer of nodes in the tree. If there are multiple circuits with the maximum number of reusable pairs, we randomly select one to proceed to the next layer. This node generation approach helps avoid generating nodes with a low probability of leading to a high-quality width-optimized solution. This node generation process is repeated until no new nodes can be generated in the l-th layer of the tree. The width optimization process ends, and the width reaches the minimum value $n - l + 1$ by using the identification method for reusable qubit pairs and the RaQCT in Section 2.2.

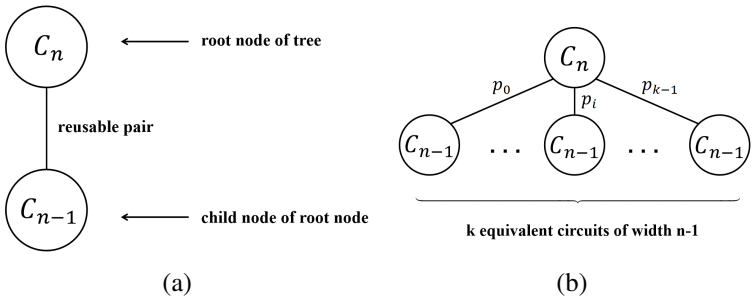


Figure 8. Width optimization of quantum circuit based on tree structure: (**a**) Viewing the circuit width optimization as a process of generating tree nodes, (**b**) Generating k child nodes by k reusable pairs.

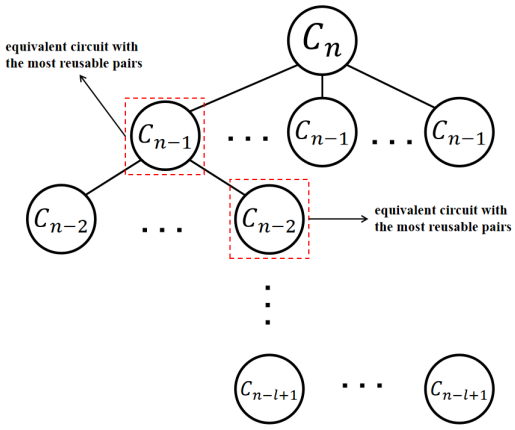


Figure 9. Width optimization process of medium or large circuits.

3.2. Width Optimization Algorithm for Quantum Circuit

Based on the above analysis, when the user seeks to obtain an exact solution for width optimization, the circuit width is optimized through hierarchical branching. In each layer, the algorithm sequentially explores all possible reusable pairs for each node, then generates the next layer of nodes, until no reusable pairs remain for the nodes in the current layer. This algorithm can search all possible width optimization solutions and obtain the exact solution for width optimization. Hence, the algorithm for obtaining the exact solution for circuit width optimization is referred to as ES.

As the scale of the circuit increases, the time required to obtain a width-optimized solution will rise significantly if the ES algorithm continues to be employed. To efficiently obtain the width optimization solution(smaller width), a greedy heuristic algorithm is adopted. The original circuit is treated as the root node of the width optimization tree. In each layer of the tree, the circuit with the highest number of reusable pairs is selected as the optimal node of the current layer, and the optimal node of the current layer is then selected to enter the next layer of the tree. Through this greedy heuristic algorithm, nodes that are less likely to lead to a high-quality width optimization solution can be pruned at each layer, thus quickly searching for a high-quality width optimization solution for the circuit. This

approach provides a locally optimal solution for the width optimization and approximates the optimal solution as soon as possible. This algorithm for obtaining the greedy heuristic solution of circuit width optimization is referred to as GHS.

Below is the pseudocode for the quantum circuit width optimization Algorithm 2:

Algorithm 2: Width Optimization of Quantum Circuit

Input: Original Candidate Circuit List *Cir_List*=[*circuit*]
Output: Optimized Width *width*

```
if TargetMethod is ES then
    while True do
        All_List ← [ ]; // Candidate circuits of next loop
        foreach cir in Cir_List do
            RP_List ← reusable_pair(cir); // List of reusable pairs
            foreach pair in RP_List do
                new_cir ← cir_transformation(pair); // Width optimization
                add new_cir to All_List;
            end
        end
        if All_List.length == 0 then
            calculate width of any new_cir in All_List;
            break; // No reuse opportunity
            return width
        end
        else
            Cir_List ← All_List; // Start the next round of width optimization
        end
    end
end
else
    while True do
        foreach cir in Cir_List do
            RP_List ← reusable_pair(cir);
        end
        if no reuse opportunity in current level then
            calculate width of any cir in Cir_List;
            break;
            return width
        end
        else
            obtain cir with max RP_List.Length from Cir_List;
            Reduce_Q ← [ ]; // Choose circuit with the most number of reusable pairs
            foreach pair in RP_List of cir do
                new_cir ← cir_transformation(pair);
                add new_cir to Reduce_Q;
            end
            Cir_List ← Reduce_Q;
        end
    end
end
```

4. Experimental Evaluation

The method proposed in this paper is compared with two state-of-the-art works [24,26]. In [24], they classified the circuits into two types: the circuits with pre-determined structures and the circuits without pre-determined structures. They defined the former as regular circuits, as they generally have a fixed gate execution sequence. In the latter, there are a large number of gates with commutativity, and the qubits operated on by these gates depend on the problem input, which is random. In [24], they handled two types of circuits by different methods. In [26], they did not propose respective methods based on different types of circuits. We assume that the method in [26] is applicable to both types of circuits.

To assess our proposed method, we also choose the two types of circuits mentioned above. The regular circuits for evaluation come from RevLib [34] and [35]. For circuits without pre-determined structures, we use the Quantum Approximate Optimization Algorithm (QAOA) [36], which is designed for solving combinatorial optimization problems on quantum computing devices [37]. The QAOA unitary is constructed by alternating between two types of operations: the mixing unitary and the problem unitary. This process is repeated for p layers, where p is a parameter that controls the depth of the algorithm. The MaxCut problem is to partition the vertices of a graph into two sets such that the number of edges between the two sets is maximized. In QAOA circuits for the MaxCut problem, each qubit represents a vertex in the graph, and the two-qubit gate connections are determined by the edges between the vertices. We chose the aforementioned circuits above because they were used for evaluation in previous works [24,26]. The method proposed in this paper is implemented in Python, and evaluated on a personal laptop with an Intel i9-13900HX CPU @2.20GHz with 16GB of RAM.

To intuitively evaluate the performance of different methods, this paper uses the following method proposed in [28] to quantify the width optimization effect of a quantum circuit:

$$e = \frac{o - f}{o}, e \in [0, 1) \tag{1}$$

where o is the original width and f is the final optimized width. The value of the e reflects the width optimization effect of the circuit. The larger the value of e, the better a width optimization method performs in obtaining high-quality width optimization solution (smaller width). Therefore, e is actually the width optimization rate of a single circuit.

For regular circuits, we evaluate our proposed width optimization methods along with those from [24,26]. We perform the GHS algorithm 5 times and record the best result during testing. Figure 10 illustrates the width optimization rate(e) achieved by applying our proposed ES and GHS to optimize the width of regular circuits, as well as a comparison with the width optimization rate(e) obtained by methods in [24,26].

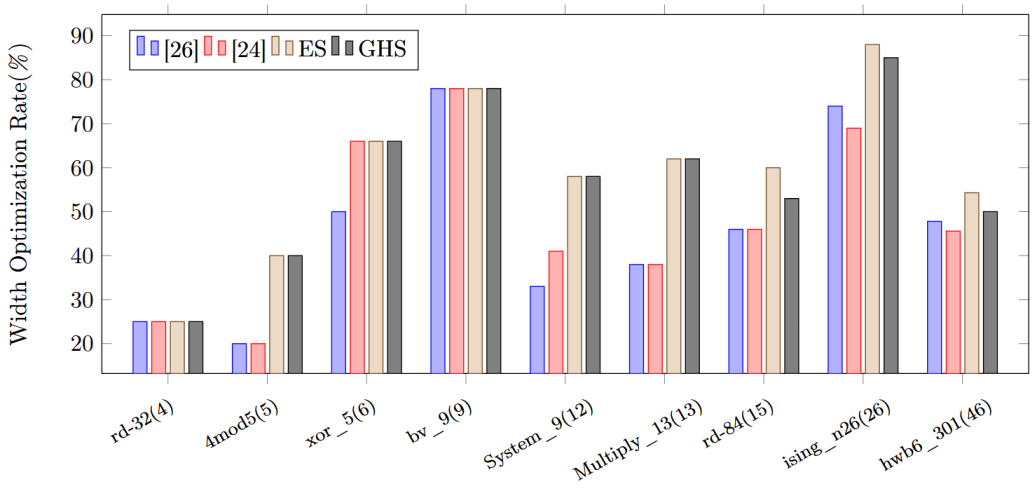


Figure 10. The width optimization rate of different regular circuits by methods in [24,26] and our proposed ES and GHS. The labels on the x-axis are the circuit names, with the original width of the circuits in parentheses on the right. The ticks on the y-axis represent the width optimization rate.

For regular circuits, both our proposed ES method and GHS method demonstrated improvements in most circuits. Specifically, our ES achieved an average width optimization rate that surpassed the method in [26] by 16.1% and the method in [24] by 11.4% across circuits for evaluation. The average width optimization rate of GHS exceeded that of [26] by 14.5% and that of [24] by 9.8%. We analyze this improvement from the following aspects.

The key point lies in the difference in the identification of reusable qubits. The method in [24] focuses on imposing additional dependencies on the gates in the circuit to determine whether the gates acting on qubits to be reused and replaced satisfy the reuse condition. The method in [26] emphasizes analyzing the causal relationships of the output qubits in the circuit to select the measurement order of the qubits. Both of their methods could achieve reusable pairs. However, they did not additionally consider adjusting the execution sequence of quantum gates in the circuit to obtain more reusable pairs. Our RaQCT can transform non-reusable qubits into reusable ones by using exchange rules of gate execution sequence in some cases. This allows us to convert qubit pairs that do not meet the reuse conditions into reusable qubit pairs. Since we apply this additional optimization at each level of the tree structure, it naturally leads to improved width optimization results, especially when a circuit can no longer be further optimized in terms of width.

For circuits without pre-determined structures, we use QAOA circuits for the MaxCut problem with p=*1* to evaluate our method and those from [24,26]. Experiments are executed on the random graphs [38] with a density of 30%. In the graph, there is a 30% probability that an edge exists between each pair of vertices, and approximately 30% of all possible edges are present in the graph. During evaluation, we use *NetworkX* [39] to generate 10 random graphs with a fixed number of vertices for tests. For our GHS, we run it 5 times and record the best result. We calculate the average width optimization rate for each method on QAOA circuits with a fixed number of qubits by:

$$ae = 1 - \frac{\sum_{i=1}^{n} f_i}{n \times m} \tag{2}$$

where n represents the number of generated random graphs with a fixed number of nodes, and m represents the fixed number of qubits in the QAOA circuit, f_i is the optimized width of the QAOA circuit with a fixed number of qubits when solving the MaxCut problem of the *i-th* random graph. Thus, *ae* denotes the average width optimization rate for QAOA circuits with a fixed number of qubits. A larger value of *ae* indicates a better average width optimization outcome(smaller width).

Figure 11 demonstrates the average width optimization rate obtained by using the methods in [24,26] and with our proposed methods. The x-axis represents QAOA circuits with a fixed number of qubits. For example, QAOA-5 means QAOA circuits with five qubits. Our proposed ES and GHS performed better than the methods from [24,26] in most circuits. Compared to the average width optimization rate of 65.2% for the method in [26] and 67.6% for the method in [24] for addressing circuits without pre-determined structures, the ES achieved an average width optimization rate of 71.6% and the GHS obtained 68.9%. In the evaluation of QAOA circuits, the effect of our method is relatively similar to the method in [24] for addressing circuits without pre-determined structures. Our method, as well as the method in [24], achieves a better width optimization rate compared to the method in [26]. The main reason is that the method in [26] did not take the exchange rules of gates into account. Actually, a large number of gates that can exchange execution sequences are in the QAOA circuits that do not have pre-determined structures. Our RaQCT, as well as the method in [24] for addressing circuits without pre-determined structures, makes full use of the exchange rules. Fully utilizing the exchange rules of gate execution sequences can enhance the width optimization rate. It is worth noting that the combination of our reusable pair identification method and RaQCT is independent of circuit types, making it more general compared to the approach presented in [24].

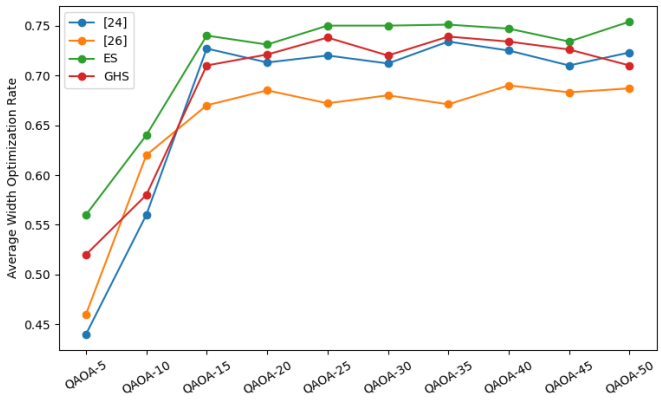


Figure 11. Comparison of average width optimization rate of QAOA circuits.

Figure 12 shows a comparison of the time costs for the ES and the GHS. The time cost is presented as the average time. The original width of a circuit around ten appears to be a dividing line. For circuits with original width smaller than ten, both ES and GHS have relatively small search trees, so the time required to obtain optimized width is relatively short. The reason GHS requires relatively less time than ES is that GHS selects a locally optimal equivalent circuit at each level to enter the next search layer. However, for circuits with original width bigger than ten, time cost of both ES and GHS starts to increase dramatically as the circuit width grows. One of the reasons is the need to evaluate more qubit pairs. As the circuit scale increases, the workload of RaQCT also begins to rise. On the other hand, the expansion of the search tree is the other reason. The time required to obtain the optimized width by GHS is less than ES. This is again due to the nature of GHS to select locally optimal solutions, making its search tree much smaller than ES, which performs enumeration. Therefore, considering the time cost, circuits with width smaller than ten are more suitable for obtaining the exact optimized width within a reasonable time range by ES. The circuits with width bigger than ten are better optimized using GHS to obtain an approximately limited optimized width.

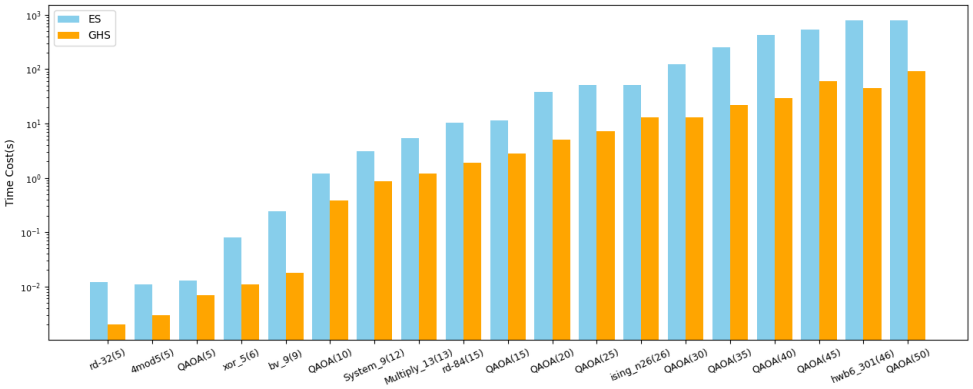


Figure 12. Comparison of time cost between ES and GHS. The labels on the x-axis represent the circuit names, and the original width of each circuit is given in parentheses.

5. Conclusion

This paper proposes a general method for identifying reusable qubit pairs from the perspective of quantum gate execution sequence. Then, the RaQCT algorithm is proposed. By performing equivalent transformations on the quantum circuit, the algorithm converts non-reusable qubit pairs into reusable ones, thereby enabling the circuit to achieve further width optimization. The reusable pair identification method and RaQCT proposed in this work exhibit greater generality, as they do not require differentiation among types of quantum circuits. Building upon this, a tree-structured width optimization model is introduced in this paper, along with both exact and heuristic algorithms for width optimization. Compared with existing works, the general framework proposed in this paper can further reduce the number of qubits required for circuit execution; these strategies further reduce the number of qubits required for circuit execution.

Although the work presented in this paper can improve the circuit width optimization performance to a certain degree, further improvements are still needed. For example, the combination of RaQCT with our width optimization strategy incurs a certain amount of time as a cost. In the future, it will be crucial to explore more scalable strategies. Optimizing circuit width is the direct benefit brought by qubit reuse for circuit optimization. However, qubit reuse can also bring other advantages and disadvantages to circuit execution [27]. Considering that optimized circuits require qubit mapping [21,23,40], the trade-offs associated with qubit reuse need to be further evaluated. In addition, some previous works [26,28] have explored and analyzed quantum algorithms suitable for circuit optimization via qubit reuse. They provided theoretical proof that some quantum algorithms can achieve optimal width through compilation utilizing qubit reuse. For example, the Bernstein-Vazirani (BV) algorithm has been theoretically proven to implement any n-qubit BV algorithm using 2 qubits through qubit reuse. In the future, as we combine qubit reuse with qubit mapping, further investigation of circuits that can benefit from qubit reuse will be required. Moreover, a thorough analysis and exploration of qubit mapping methods that incorporate qubit reuse will be conducted, considering the specific characteristics of these circuits.

Author Contributions
H.T. provides conceptualization and methodology of the article. H.T. designs and performs the experiments. H.T. writes the article. Z.G., X.C., F.D. and S.Z. participate in writing comments and process supervision. Z.G. is responsible for project management and providing financial support. All authors have read and agreed to the published version of the manuscript.

Funding
This work was supported in part by the National Natural Science Foundation of China (No. 62072259), in part by the Natural Science Foundation of Jiangsu Province (No. BK20221411), in part by the PhD Start-up Fund of Nantong University (No. 23B03), in part by Science and Technology Program for Social Welfare and People's Livelihood of Nantong (No. MS2023070).

Conflicts of Interest
The authors declare no conflicts of interest.

Data Availability Statement
All data generated or used during the study appear in the submitted article.

References

1. David Peral-García, Juan Cruz-Benito, and Francisco José García-Peñalvo. Systematic literature review: Quantum machine learning and its applications. *Computer Science Review*, 51:100619, 2024.
2. Ubaid Ullah and Begonya Garcia-Zapirain. Quantum machine learning revolution in healthcare: a systematic review of emerging perspectives and applications. *IEEE Access*, 2024.
3. Stepan Fomichev, Kasra Hejazi, Modjtaba Shokrian Zini, Matthew Kiser, Joana Fraxanet, Pablo Antonio Moreno Casares, Alain Delgado, Joonsuk Huh, Arne-Christian Voigt, Jonathan E Mueller, et al. Initial state preparation for quantum chemistry on quantum computers. *PRX Quantum*, 5(4):040339, 2024.
4. Kieran Dalton, Christopher K Long, Yordan S Yordanov, Charles G Smith, Crispin HW Barnes, Normann Mertig, and David RM Arvidsson-Shukur. Quantifying the effect of gate errors on variational quantum eigensolvers for quantum chemistry. *npj Quantum Information*, 10(1):18, 2024.
5. Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068: 1–66, 2024.
6. Muhammad AbuGhanem and Hichem Eleuch. Nisq computers: a path to quantum supremacy. *IEEE Access*, 2024.
7. Sukhpal Singh Gill, Oktay Cetinkaya, Stefano Marrone, Elias F Combarro, Daniel Claudino, David Haunschild, Leon Schlote, Huaming Wu, Carlo Ottaviani, Xiaoyuan Liu, et al. Quantum computing: Vision and challenges. *arXiv preprint arXiv:2403.02240*, 2024.
8. Marcello Caleffi, Michele Amoretti, Davide Ferrari, Jessica Illiano, Antonio Manzalini, and Angela Sara Cacciapuoti. Distributed quantum computing: a survey. *Computer Networks*, 254:110672, 2024.
9. Christophe Piveteau and David Sutter. Circuit knitting with classical communication. *IEEE Transactions on Information Theory*, 2023.
10. Shisheer S Kaushik, M Thirumalai, and Harsh Mehta. From static to dynamic: Implementation of long-range entanglement ghz states for dynamic circuit-based quantum teleportation. In *2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6. IEEE, 2024.
11. Antonio D Córcoles, Maika Takita, Ken Inoue, Scott Lekuch, Zlatko K Minev, Jerry M Chow, and Jay M Gambetta. Exploiting dynamic quantum circuits in a quantum algorithm with superconducting qubits. *Physical Review Letters*, 127(10):100501, 2021.
12. Juan M Pino, Jennifer M Dreiling, Caroline Figgatt, John P Gaebler, Steven A Moses, MS Allman, CH Baldwin, Michael Foss-Feig, David Hayes, Karl Mayer, et al. Demonstration of the trapped-ion quantum ccd computer architecture. *Nature*, 592(7853):209–213, 2021.
13. Riddhi S Gupta, Ewout Van Den Berg, Maika Takita, Diego Riste, Kristan Temme, and Abhinav Kandala. Probabilistic error cancellation for dynamic quantum circuits. *Physical Review A*, 109(6):062617, 2024.
14. Talal Ahmed Chowdhury, Kwangmin Yu, Mahmud Ashraf Shamim, ML Kabir, and Raza Sabbir Sufian. Enhancing quantum utility: simulating large-scale quantum spin chains on superconducting quantum computers. *Physical Review Research*, 6(3):033107, 2024.

15. Ivan Pogorelov, Thomas Feldker, Ch D Marciniak, Lukas Postler, Georg Jacob, Oliver Krieglsteiner, Verena Podlesnic, Michael Meth, Vlad Negnevitsky, Martin Stadler, et al. Compact ion-trap quantum computing demonstrator. *PRX Quantum*, 2(2):020343, 2021.
16. Elisa Bäumer, Vinay Tripathi, Alireza Seif, Daniel Lidar, and Derek S Wang. Quantum fourier transform using dynamic circuits. *Physical Review Letters*, 133(15):150602, 2024.
17. Sohrab Sajadimanesh and Ehsan Atoofian. Implementation of a quantum division circuit on noisy intermediate-scale quantum devices using dynamic circuits and approximate computing. *Physical Review A*, 109(5):052601, 2024.
18. VV Sivak, Alec Eickbusch, Baptiste Royer, Shraddha Singh, Ioannis Tsioutsios, Suhas Ganjam, Alessandro Miano, BL Brock, AZ Ding, Luigi Frunzio, et al. Real-time quantum error correction beyond break-even. *Nature*, 616(7955):50–55, 2023.
19. Movahhed Sadeghi, Soheil Khadirsharbiyani, and Mahmut Taylan Kandemir. Quantum circuit resizing. *arXiv preprint arXiv:2301.00720*, 2022.
20. Leo Sünkel, Darya Martyniuk, Denny Mattern, Johannes Jung, and Adrian Paschke. Ga4qco: genetic algorithm for quantum circuit optimization. *arXiv preprint arXiv:2302.01303*, 2023.
21. Gushu Li, Yufei Ding, and Yuan 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*, pages 1001–1014, 2019.
22. Pengcheng Zhu, Zhijin Guan, and Xueyun Cheng. A dynamic look-ahead heuristic for the qubit mapping problem of nisq computers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4721–4735, 2020.
23. Siyuan Niu, Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. A hardware-aware heuristic for the qubit mapping problem in the nisq era. *IEEE Transactions on Quantum Engineering*, 1:1–14, 2020.
24. Fei Hua, Yuwei Jin, Yanhao Chen, Suhas Vittal, Kevin Krsulich, Lev S Bishop, John Lapeyre, Ali Javadi-Abhari, and Eddy Z Zhang. Caqr: A compiler-assisted approach for qubit reuse through dynamic circuit. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 59–71, 2023.
25. Alexandru Paler, Robert Wille, and Simon J Devitt. Wire recycling for quantum circuit optimization. *Physical Review A*, 94(4):042337, 2016.
26. Matthew DeCross, Eli Chertkov, Megan Kohagen, and Michael Foss-Feig. Qubit-reuse compilation with mid-circuit measurement and reset. *Physical Review X*, 13(4):041057, 2023.
27. Sebastian Brandhofer, Ilia Polian, and Kevin Krsulich. Optimal qubit reuse for near-term quantum computers. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 1, pages 859–869. IEEE, 2023.
28. Kun Fang, Munan Zhang, Ruqi Shi, and Yinan Li. Dynamic quantum circuit compilation. *arXiv preprint arXiv:2310.11021*, 2023.
29. Siyuan Niu, Akel Hashim, Costin Iancu, Wibe Albert De Jong, and Ed Younis. Effective quantum resource optimization via circuit resizing in bqskit. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.
30. Hanru Jiang. Qubit recycling revisited. *Proceedings of the ACM on Programming Languages*, 8(PLDI):1264–1287, 2024.
31. Gideon Uchehara, Tor M Aamodt, and Olivia Di Matteo. Graph-based identification of qubit network (gidnet) for qubit reuse. *arXiv preprint arXiv:2410.08817*, 2024.
32. Zhijin Guan, Renjie Liu, Xueyun Cheng, Shiguang Feng, and Pengcheng Zhu. Suppression of crosstalk in quantum circuit based on instruction exchange rules and duration. *Entropy*, 25(6):855, 2023.
33. Yan Ge, Wu Wenjie, Chen Yuheng, Pan Kaisen, Lu Xudong, Zhou Zixiang, Wang Yuhan, Wang Ruocheng, and Yan Junchi. Quantum circuit synthesis and compilation optimization: Overview and prospects. *arXiv preprint arXiv:2407.00736*, 2024.
34. Robert Wille, Daniel Große, Lisa Teuber, Gerhard W Dueck, and Rolf Drechsler. Revlib: An online resource for reversible functions and reversible circuits. In *38th International Symposium on Multiple Valued Logic (ismvl 2008)*, pages 220–225. IEEE, 2008.
35. Ang Li, Samuel Stein, Sriram Krishnamoorthy, and James Ang. Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation. *ACM Transactions on Quantum Computing*, 4(2):1–26, 2023.

36. Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020.
37. Zeqiao Zhou, Yuxuan Du, Xinmei Tian, and Dacheng Tao. Qaoa-in-qaoa: solving large-scale maxcut problems on small quantum machines. *Physical Review Applied*, 19(2):024027, 2023.
38. Sourav Chatterjee and SR Srinivasa Varadhan. The large deviation principle for the erdős-rényi random graph. *European Journal of Combinatorics*, 32(7):1000–1017, 2011.
39. Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.
40. Di Yu and Kun Fang. Symmetry-based quantum circuit mapping. *Physical Review Applied*, 22(2):024029, 2024.