axioms

MDPI

Article

Quantum Circuit Template Matching Optimization Method for Constrained Connectivity

Xiaofeng Gao [1], Zhijin Guan [1,*], Shiguang Feng [2] and Yibo Jiang [1]

[1] School of Information Science and Technology, Nantong University, Nantong 226001, China; 2110310048@stmail.ntu.edu.cn (X.G.)
[2] School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510006, China
* Correspondence: guan.zj@ntu.edu.cn

Abstract: The execution of quantum algorithms requires two key considerations. On the one hand, it should meet the connectivity constraint requirements of quantum circuit mapping for quantum architectures, and on the other hand, it needs to consider reducing the probability of errors in the execution of quantum circuits as much as possible. This paper proposes a novel optimization technique based on template matching that to satisfy both requirements. The template matching optimization method can significantly reduce the number of gates in a quantum circuit and further enhance its practicality. It stands as advanced optimization technology available today. Our method optimizes quantum logic circuits mapped onto quantum architecture by initially selecting their linear substructure. We then zone the circuit according to the gate dependency graph and optimize each block through template matching. Finally, we reorganize the circuit to obtain the optimized version as the final result. Our proposed method is amenable to various quantum architectures. To evaluate its efficacy, we conduct a comparative analysis with the t|ket⟩ and Qiskit compiler using a set of benchmark test circuits. Specifically, compare to the t|ket⟩ compiler method, the highest average optimization rate of our method can reach 25.75%. Compare with the Qiskit compiler method, the highest average optimization rate can reach 32.72%. Overall, our approach has significant optimization advantages.

Keywords: quantum circuit optimization; template matching; linear substructure

MSC: 81P68

Citation: Gao, X.; Guan, Z.; Feng, S.; Jiang, Y. Quantum Circuit Template Matching Optimization Method for Constrained Connectivity. *Axioms* **2023**, *12*, 687. https://doi.org/10.3390/axioms12070687

Academic Editor: Julio Lopez-Saldivar

Received: 7 June 2023
Revised: 8 July 2023
Accepted: 12 July 2023
Published: 14 July 2023

1. Introduction

Noisy Intermediate-Scale Quantum (NISQ) computers [1] have the potential to increase computational power for certain problem classes, with tens to hundreds of qubits. However, limitations exist concerning the number of qubits and decoherence [2], which degrades the quantum information stored in the qubits over time. These challenges restrict the number of quantum gates that can be applied to quantum circuits, and limit the complexity of computations that can be executed on quantum devices. Consequently, it is imperative to optimize quantum circuits executed on NISQ devices. A key method to optimization is to minimize the number of quantum gates by replacing or eliminating some gates in the circuit. In the long run, optimizing quantum circuits not only improves running time for quantum algorithms but also plays an essential role in ensuring high-quality operation of quantum devices in the near future.

Currently, most quantum circuit optimization methods assume that the circuit will be executed on a general-purpose quantum computer, regardless of the connectivity of the actual hardware structure. Examples of circuit optimization methods can be found in sources such as [3–6], which do not take into account the connectivity constraints of quantum computer. In order for a quantum logic circuit to operate on a quantum device, circuit mapping needs to be performed by inserting SWAP gates to enable any number of

quantum gates in the presence of connection constraints. However, circuit mapping often results in excessive extra gates and increased circuit depth. Although efficient mapping methods [7–11] have been developed to generate physical quantum circuits that require as few gates as possible, there is still space for further optimization of mapping circuits. In order to avoid changing the connection constraints that the circuit has already satisfied, some optimization methods, such as the t|ket⟩ compiler method [12], simplify the mapping circuit only by eliminating adjacent reverse gates, so this optimization effect is minimal.

As quantum computing continues to advance at a rapid pace, it becomes increasingly critical to develop more efficient methods for optimizing quantum circuits. To address this challenge, a novel quantum circuit optimization method is proposed in this paper, which takes into account the practical constraints of quantum architecture connections. By doing so, this method minimizes the number of quantum gates used while ensuring that the quantum architecture connections are correctly accounted for, resulting in a significant improvement in the overall performance of mapped quantum circuit.

In this paper, we extend the template matching optimization method described in the reference [3]. We introduce the concept of linear templates and utilize them for the purpose of matching optimization. Our goal is to optimize the mapped quantum circuit while maintaining its connectivity constraints. To achieve this goal, we propose several methods including linear substructure selection, quantum circuit zoning, zoning circuit optimization and reorganization. These techniques enable us to further optimize quantum circuits and enhance the performance of circuit operations on quantum devices.

Before optimization, existing templates are refactored by adding SWAP gates or replacing Bridge gates to fit the connectivity constraints of the linear topology. The optimization comprises four primary steps: (1) the identification of linear substructures based on the number of Controlled-NOT(CNOT) gates between qubits that are mapped to the quantum topology; (2) the zoning of circuits according to the selected linear substructures and based on the gate dependency graph; (3) the optimization of each quantum circuit zoning using block-by-block template matching optimization techniques; (4) reorganization of optimized circuits based on the original zoning rules to generate new circuits equivalent to the original but with reduced gates. To optimize all qubits in the quantum topology, the above steps are repeated several times.

2. Preliminaries

In this section, we provide the fundamental definitions and notations related to quantum circuits.

2.1. Quantum Gate and Quantum Circuit

Quantum computing [13] provides a computing paradigm based on quantum bits. Quantum bits can not only represent Boolean 0 and Boolean 1, but also represent the superposition of the two. A quantum bit $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ such that $|\alpha|^2 + |\beta|^2 = 1$. If $\alpha = 1$, then $|\varphi\rangle$ represents the classical 0; if $\beta = 1$, then $|\varphi\rangle$ represents the classical 1. Quantum gates are used to operate on qubits in quantum computer.

A quantum circuit is a computational model consisting of quantum gates as the basic elements. These gates are operations selected from a quantum gate library [14], which is a collection of quantum gates capable of implementing any reversible function. A quantum logic circuit can be represented as a network structure diagram composed of a cascade of qubits and quantum gates. The circuit can also be represented by a unitary matrix, which is calculated as the product of matrices representing individual gates. Each horizontal line in the circuit diagram represents a qubit, and n parallel horizontal lines represent n qubits recorded as $q_0, q_1, \ldots, q_n$ from the top down. Quantum gates execute sequentially according to their position in the circuit from left to right, recorded as $g_0, g_1, \ldots, g_m$. A CNOT circuit is defined as a quantum circuit that is only cascaded by CNOT gates. Figure 1 shows a CNOT circuit with 5 CNOT gates.

Due to the higher error rate of two-qubit gates [1], including CNOT gates, compared to single-qubit gates in NISQ devices, and the consideration of the nearest neighbor between the control and target bits, this paper solely focuses on optimizing CNOT gates for quantum circuit optimization.

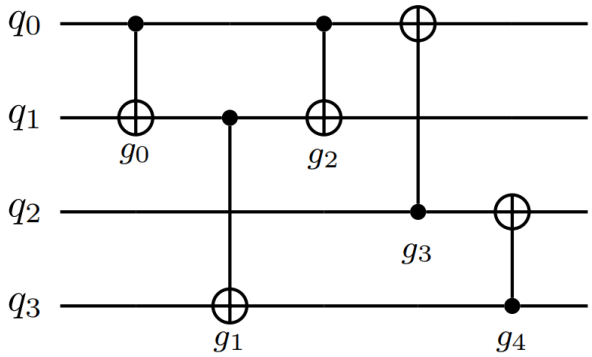

Figure 1. Quantum circuit diagram.

2.2. Quantum Cost

The number of gates in a quantum circuit and the number of physical operations required to achieve each gate are important factors in measuring computational efficiency and computational success, hence the name quantum cost [15]. Interference from the external environment can lead to the decoherence of quantum systems, so quantum computing must be completed in a finite coherence time. This requires that the quantum cost of quantum circuits be minimized, and the circuit with the smallest quantum cost to achieve a particular function is called an optimal circuit [16–18]. Although finding the optimal circuit is a QMA (Quantum Merlin Arthur) problem [19], various practical methods for simplifying quantum circuits (such as [20–22]) have been proposed to reduce the cost of implementation. If the number of quantum gates is used as the quantum cost standard, a CNOT gate represents a quantum cost of 1.

2.3. Quantum Topology

The topology of a quantum architecture represents the connectivity between qubits in a quantum computing device. To ensure that a quantum logic circuit can be executed on a quantum computing device, it must satisfy the connectivity constraints of the quantum topology. There are currently many types of quantum architectures, including ibmq_montreal, ibm_perth, Rigetti Agave, ibmq_guadalupe, etc. [23]. Figure 2a,b show the topology of ibm_perth and ibmq_guadalupe, respectively.

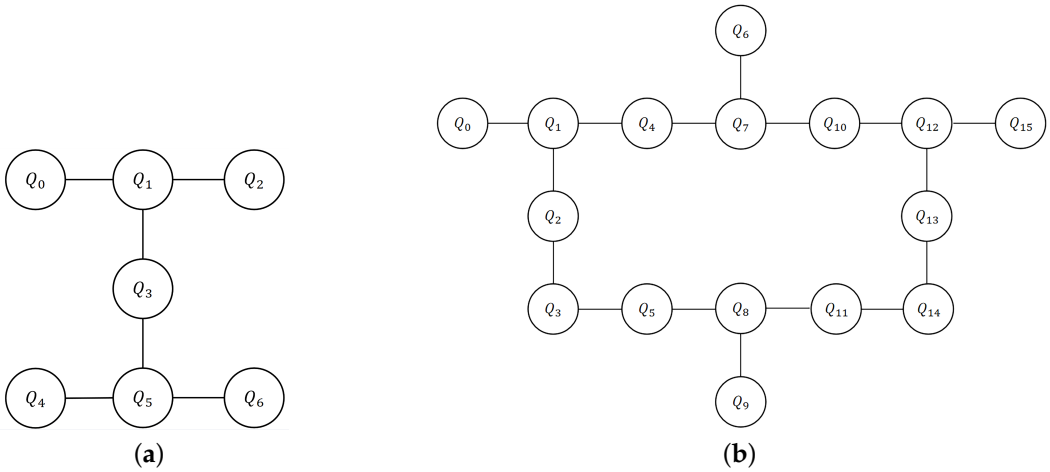

Figure 2. Quantum topology. (**a**) Ibm_perth. (**b**) Ibmq_guadalupe.

2.4. Quantum Circuit Mapping

When executing a quantum circuit on a quantum computer, the qubits in the logic circuit need to be mapped to physical qubits on the target quantum architecture. For certain constraints of quantum architectures, such as superconducting quantum architectures, when mapping two qubit gates, their control and target qubits must map to physically connected adjacent qubits. Due to the limited connectivity of physical qubits on current devices, it is generally considered difficult to find an initial quantum graph satisfying all double qubit gates in the entire circuit. Therefore, mapping algorithms and optimization strategies are needed to address this limitation and find the best mapping of physical qubits, which may involve additional qubit gates to compensate for the lack of connectivity [7–11]. In this paper, the mapping quantum circuit is optimized considering the connectivity constraint, so a better mapping method is used to obtain the experimental quantum circuit.

We compare IBM Qiskit [24] with the t|ket⟩ compiler [12], and for the original quantum circuit, this experiment uses the general optimization method in the t|ket⟩ compiler to optimize the circuit and perform qubit mapping, which can efficiently produce shorter circuit.

2.5. Quantum Circuit Templates

Definition 1. *If a quantum logic circuit T consists of a series of unitary quantum gates U_i satisfying the gate unitary matrix multiplication $U_{|T|} \cdots U_1 = 1$ (where $|T|$ is the number of gates in the template), then the circuit T is called the template [25]. Figure 3 shows a CNOT gate template.*

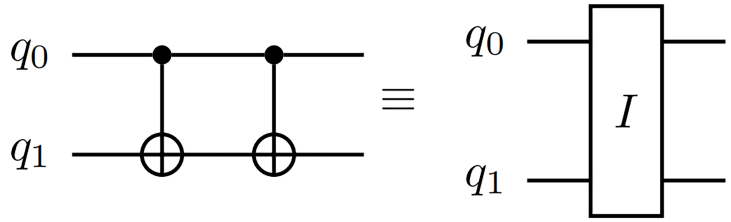

Figure 3. Circuit template example.

Template matching optimization: It is assumed that the gate sequence $U_a \cdots U_b (1 \leq a \leq b \leq |T|)$ is found in the circuit C, which matches a set of gate sequences in the template T. According to Definition 1, a series of gates in the template satisfy $U_{|T|} \cdots U_1 = 1$, and each gate U_i has an inverse gate $U_i^\dagger$, then the gate sequence $U_a \cdots U_b$ can be equivalently expressed as $U_{a-1}^\dagger \cdots U_1^\dagger U_{|T|}^\dagger \cdots U_{b+1}$. If the quantum cost of $U_{a-1}^\dagger \cdots U_1^\dagger U_{|T|}^\dagger \cdots U_{b+1}$ gate sequence is lower than that of $U_a \cdots U_b$, then we can reduce the number of quantum gates in circuit C by substitution. That is to say, in order to reduce more quantum gates, the longer the matching gate sequence $U_a \cdots U_b$, the better [3].

The template library contains all the different templates, and the templates used in this article are mainly referenced in Qiskit [24].

2.6. Gate Dependency Graph

The gate dependency graph [26] is a representation of quantum circuits. First, the vertices in the graph correspond to the individual gates in the circuit. The graph has an edge $i \to j$ from vertex i to vertex j if by repeatedly interchanging commuting gates in the circuit, one can bring gate i immediately to the left of gate j, but gates i and j themselves do not commute. In other words, in the commuted circuit gate j immediately follows gate i, but one cannot change the order of gates i and j [3]. The quantum gate g_0 in Figure 4a, g_1, g_2 can be exchanged with each other to obtain the quantum circuit as shown in Figure 4b, and the quantum circuit of (a) and (b) is equivalent. From this, a gate dependency graph representation can be obtained as shown in Figure 4c.

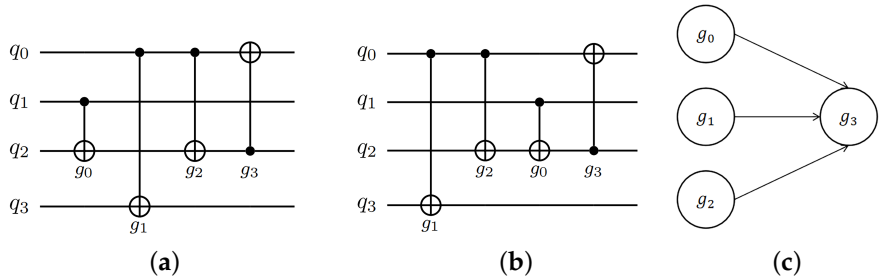

Figure 4. Different equivalent representations of quantum circuits. (**a**) Example of a quantum circuits. (**b**) The circuit is modified by interchanging the order of commuting gates. (**c**) Gate dependency graph.

3. Selection of Linear Substructure

When mapping quantum logic circuits to quantum architectures, it is important to consider the connectivity constraints of the quantum architectures. There are many types of quantum architectures, but not all of the existing templates may be suitable for meeting the connectivity requirements of quantum architectures. If an existing template is used for matching optimization directly, the resulting optimized circuit may not satisfy the connectivity constraints of the quantum architectures. Therefore, it is essential to carefully consider the specific architecture and connectivity constraints of the target architecture when mapping quantum circuits to quantum architectures.

For example, Figure 5a is a known quantum logic circuit, it conforms to the connectivity constraint of the linear architecture. Figure 5b is a known template. If the quantum gates g'_1, g'_2, g'_3, g'_4 are matched with the gates g_3, g_4, g_5, g_6 in the circuit of Figure 5a, the matching gates g_3, g_4, g_5, g_6 in the circuit are replaced with the gate g'_5 in the template, and the circuit is optimized as shown in Figure 5c. It can be seen that there is a non-nearest neighbor gate g_3 in the optimized circuit Figure 5c, which does not conform to the connectivity constraint of the linear architecture.

To address the aforementioned issue and find a solution that can be applied to all architectures, it is necessary to identify a common substructure that exists in all architectures. By ensuring that the template satisfies the connectivity of this substructure, it is possible to optimize the circuit without altering the connectivity constraints of the circuit. In this way, the resulting optimized circuit will be compatible with the connectivity requirements of the target quantum architecture, without compromising the functionality of the original circuit. It is therefore crucial to identify and leverage common substructures when mapping quantum circuits to quantum architectures.

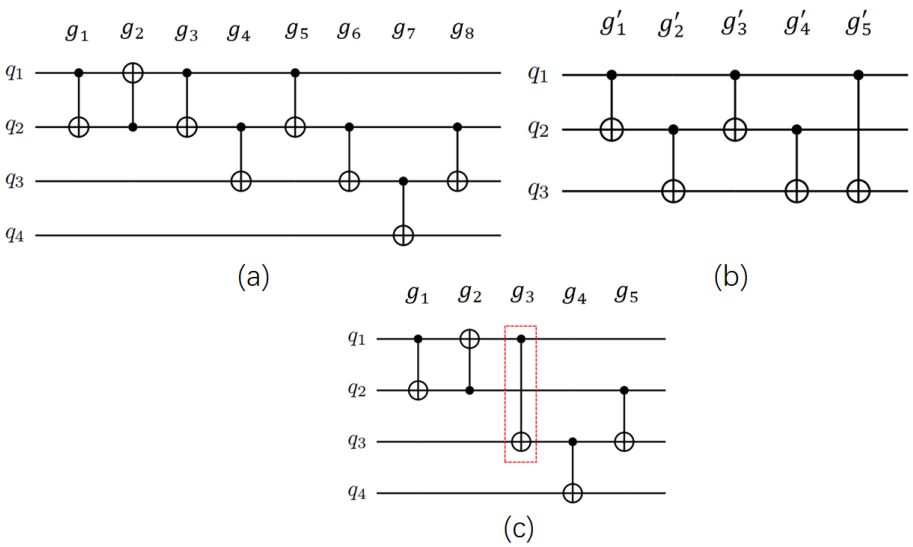

Figure 5. Template matching optimization. (**a**) Proximity quantum circuits. (**b**) Template circuit. (**c**) Template matching optimized circuit.

Definition 2. *A quantum linear topological structure is a type of quantum network topology where the qubits are arranged in a linear chain or one-dimensional array, and the connections between them are limited to nearest-neighbor interactions.*

Quantum linear topology is generally a part of the topology of quantum computing system, that is, quantum linear topology is a substructure of the topology of quantum computing system. As shown in Figure 2 ibm_perth (Q_0, Q_1, Q_2) and (Q_0, Q_1, Q_2, Q_3, Q_5) in Figure 2 ibmq_guadalupe are the linear topology of this quantum topology.

Definition 3. *A template is called a linear template if it satisfies the connectivity constraint of quantum linear topology.*

For templates in a given template library, templates in the library that do not satisfy the linear topology can be reconstructed to generate a new template library. Template reconstruction method is mainly: first, find the template in the non-neighbor CNOT gate; then, by inserting the SWAP gate or introducing the Bridge gate, the non-neighboring CNOT gate is neighbored to satisfy the topological connectivity. Finally, a new template conforming to the connectivity constraint of linear topology is obtained. Such reconstructed templates can perform template matching optimization on all circuits that satisfy linear topological connectivity.

Indeed, the ubiquity of linear structures in all topologies makes them a useful basis for optimizing the mapping of quantum circuits to various quantum topologies. By utilizing linear templates, we can develop techniques that are applicable across different topologies, rather than designing solutions specific to each individual topology.

Therefore, selecting the appropriate linear substructure in the quantum topology is a crucial step in achieving optimal results in quantum circuit optimization. By identifying and leveraging linear substructures in quantum topologies, we can greatly improve the effectiveness of quantum circuit optimization techniques, resulting in more efficient quantum computations.

Topological Linear Substructure Selection

When optimizing a quantum circuit by template matching, the number of qubits and quantum gates of the optimized quantum circuit should not be less than the number of qubits and quantum gates of the smallest template circuit in the template library.

As the number of qubits in a quantum circuit increases, the number of quantum gates in the circuit also increases. For a quantum circuit that contains more gates, the types and numbers of template circuits that can be matched in the template library also increase, thus allowing for better optimization.

This is because more complex quantum circuits generally require more gates to be implemented, and more gates mean that the quantum system can perform more operations and transformations. Therefore, quantum circuits with more gates are better suited to express complex operations, leading to better performance.

In addition, more complex quantum circuits can enable a greater variety of operations and logic operations, resulting in more types and numbers of template circuits that can be matched. This will facilitate a more comprehensive and detailed optimization of the quantum circuit, thus improving system reliability and implementation efficiency.

Therefore, when selecting a linear substructure, it is advisable to choose a substructure with as many qubits as possible, and ensure that the corresponding quantum logic circuit contains as many quantum gates as possible.

Suppose l qubits are selected in the topology, expressed as the set $S = Q_{s1}, \ldots, Q_{sl}$, then there are k qubits adjacent (connected) to the qubits in S, expressed as $T = Q_{t1}, \ldots, Q_{tk}$, $k \geq 0, Q_{ti} \notin S, i \in 1 \cdots k$. If $k \geq 2$, the number of quantum gates between the logic circuit qubit pairs (q_{i1}, q_{j1}) and (q_{i2}, q_{j2}) corresponding to different adjacent qubit pairs (Q_{ti1}, Q_{sj1}) and (Q_{ti2}, Q_{sj2}) composed of qubits in T and S may be different.

As shown in Figure 6, when we select the $(Q_0, Q_1, Q_3, Q_5, Q_6)$ qubits on the linear substructure in the topology, there are also two qubits, Q_2 andQ_4, adjacent to (connected) to Q_1 and Q_5, respectively. From the corresponding quantum logic circuit in Figure 7, we can see that there are CNOT gates g_4 and g_7 between q_1q_2 and q_4q_5 qubits, respectively., which are not within the upper gate of the qubit we selected. Their presence affects the number of qubits and gates of the quantum circuit when the template matching optimization.

Definition 4. *The barrier gate is a two-qubit gate with its control qubit and target qubit respectively located on the selected qubit and other qubits.*

Barrier gate g_7 prevents the merger of its successor gate g_8 with its predecessor gate g_0. When optimization, because of this barrier, the gates on the selected linear substructure cannot be directly optimized together. When there are more such barrier gates, the entire circuit is zoned into more local circuits, which reduces the number of qubits and gates on the quantum local circuit to be optimized. Therefore, if there are fewer barrier gates in a quantum circuit, more optimization can be performed through template matching using quantum local circuits containing more gates.

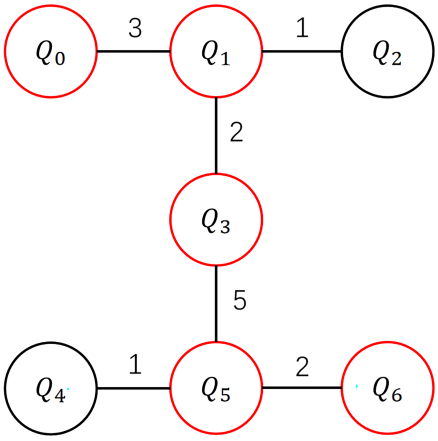

Figure 6. Linear substructure selection (selected qubits are highlighted with red circles).

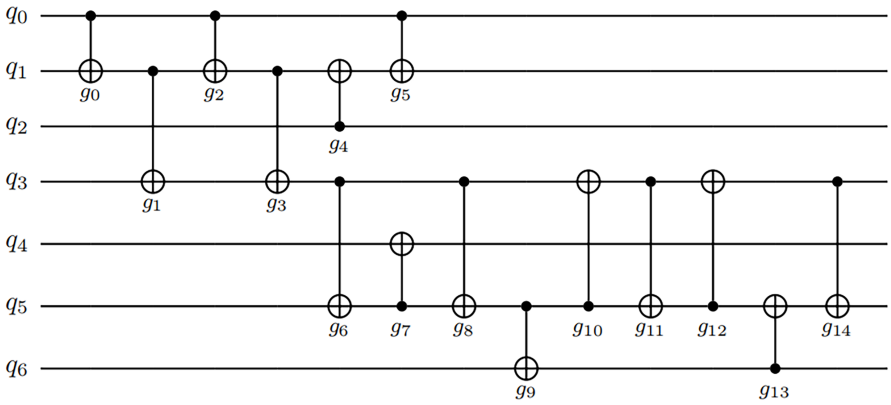

Figure 7. Example of a quantum circuit.

To satisfy template matching conditions and maximize the number of qubits and quantum gates, it's important to consider the number of quantum gates between adjacent qubits in the corresponding quantum logic circuit. In order to select the maximum number of quantum gates in the quantum logic circuit and minimize the number of barrier gates, the following qubit selection rules are applied in the topology.

In the quantum topology, for a selected qubit Q_i, if there are more than m adjacent qubits $Q_{j1}, \ldots, Q_{jm}, m \geq 2$ with Q_i, then in the quantum logic circuit corresponding to

$(q'_i, q'_{j1}), \ldots, (q'_i, q'_{jm})$, the two qubits with the highest number of gates between the two qubits are preferred. A weight α is given below, with the larger the α the better the result.

$$\text{Weight } \alpha = \frac{N_a}{N_b} \tag{1}$$

where N_a is the number of quantum gates on the chosen qubit; N_b is the number of barrier gates.

For qubit requirements, we use the outermost qubit in the topology as the starting point (as shown in Figure 2 ibm_perth Q_0, Q_2, Q_4, Q_6), and then select subsequent qubits based on the number of gates between qubits. This allows us to select as many qubits as possible while ensuring the fewest number of barrier gates. This is illustrated in Algorithm 1.

Take the ibm_perth topology in Figure 2 and the quantum circuit in Figure 7 as examples. Firstly, according to the number of gates between every two qubits, the outermost Q_0, Q_1 two qubits of the topology are selected; Then, for the successor qubits of the qubit Q_1, Q_2 and Q_3, because there are more qubits on the corresponding quantum logic circuit between Q_1, Q_3, the qubit Q_3 is selected; then, the Q_5, Q_6 are selected sequentially; finally, the linear substructure shown in the red qubit in Figure 6 is obtained.

The specific first round linear substructure selection algorithm is described as follows:

Algorithm 1: The first round of linear substructure selection algorithm

Input : Topology $G(V, E)$, the number of gates between qubits
Output: The selected substructure qubits
1. Initialize a list W to store qubits
2. for $v \in V$ *do*
3. for $e \in E$ *do*
4. if v_i *have the least adjacent connection nodes then*
5. for $v_j \in v_i$ *connection points do*
6. if e_{ij} *has maximum weight then*
7. Add $v_i v_j$ *to to the list of W*
8. else if no adjacent nodes can be selected then
9. break
10. end if
11. end for
12. end for
13. return W

4. Circuit Zoning Optimization and Reorganization

4.1. Circuit Zoning

After selecting the linear substructure, the circuit cannot be optimized directly due to the presence of other qubits that are coupled to those on the chosen substructure. The quantum gates between these qubits, also known as barrier gates, limit the number of qubits that can be included in the selected substructure. This effect is discussed in more detail in Section 3.

As shown in Figure 8, the circuit is zoned into two parts by the barrier gate g_7, a quantum circuit consisting of a gate $(g_0, g_1, g_2, g_3, g_5, g_6)$ and the other is a quantum circuit consisting of a gate $(g_8, g_9, g_{10}, g_{11}, g_{12}, g_{13}, g_{14})$. Therefore, before optimizing the circuit, it is necessary to zone the gates in the circuit through the gate dependency graph according to the limitations of the quantum gate. Then, the template matching optimization method is used to optimize the circuits of the zoning one by one.

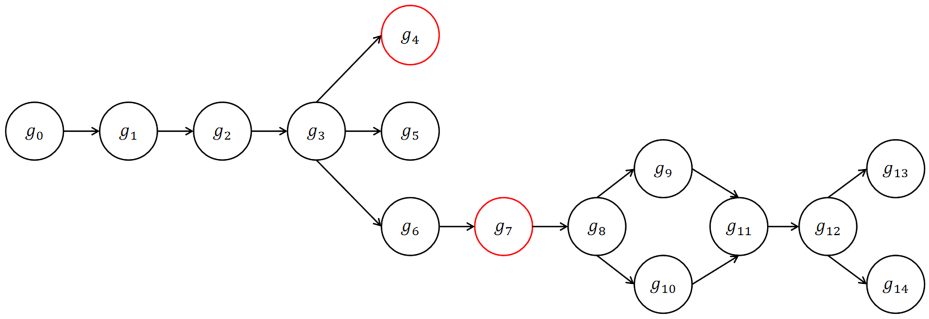

Figure 8. Gate dependence graph of quantum circuit (Figure 7).

The zoning of the circuit is mainly based on the gate dependency graph. The specific steps are as follows:

Step 1: Determine whether the first quantum gate on the quantum logic circuit corresponding to the selected qubit is a barrier gate. If it is not a barrier gate, take the first gate as the starting point and proceed to the second step. If it is a barrier gate, start with a successor gate that is not a barrier gate and proceed to the second step.

Step 2: Determine whether the successor gate of the starting point has a barrier gate. If it does, then the barrier gate and all the gates that are subsequently affected are not within the bounds of the zoning. If the other successor gates are not barrier gates, zone them together.

Step 3: Continue to determine whether the gate behind it has a barrier gate according to step 2. Repeat this process until the last zoned gate is followed only by barrier gates, and then the circuit zoning of the first part ends.

Step 4: Begin with the undivided quantum gate that is not a barrier gate and continue the zoning through steps 2 and 3.

Step 5: Repeat step 4 until the last gate on the quantum circuit is reached and the zoning is complete.

Using Figure 8 as an example, we start by selecting gate g_0 as the starting point on the qubit. The successor gate g_1 is not a barrier gate, so both g_0 and g_1 are zoned together. The successor gates g_2 and g_3 are also zoned together.

The successor gate of the g_3 gate is $g_4 g_5 g_6$, where g_4 is a barrier gate without a successor gate. However, g_5 and g_6 are not barrier gates, so they are zoned together. Since g_5 has no successor gate, the zoning is interrupted.

The successor gate of g_6 is only the barrier gate g_7, so the first part of the zoning ends here. The zoned gates produce the quantum circuit shown in Figure 9.

The second part of the zoning starts with the successor gate g_8 of the g_7 gate. As g_8 is not a barrier gate, it is zoned. The successor gates do not barrier the zoning, and we obtain the quantum circuit shown in Figure 10 using the zoned gates. At this point, all gates in the circuit are zoned. This is illustrated in Algorithm 2.

The specific zoning algorithm is described as follows:

Algorithm 2: Circuit zoning

Input : Gate dependency graph
Output: Zoning circuits
1. Initialize a list C to store gates
2. for $i \in 0, 1, \ldots, n$ *do*
3. if $j \in$ *min gates on selected qubit then*
4. Add g_j *to the list of C*
5. if one successor gate of $g_j \in$ *block gates then*
6. continue
7. else if one successor gate of $g_j \in$ *gates on selected qubit then*
8. Add the successor gate g_{j+a} *to the list of C*
9. else if the only successor gate of $g_j \in$ *block gates then*
10. break
11. end if
12. if one successor gate of $g_i \in$ *block gates then*
13. continue
14. else if one successor gate of $g_i \in$ *gates on selected qubit then*
15. Add the successor gate g_{i+a} *to the list of C*
16. else if the only successor gate of $g_i \in$ *block gates then*
17. break
18. end if
19. end if
20. end for
21. return C

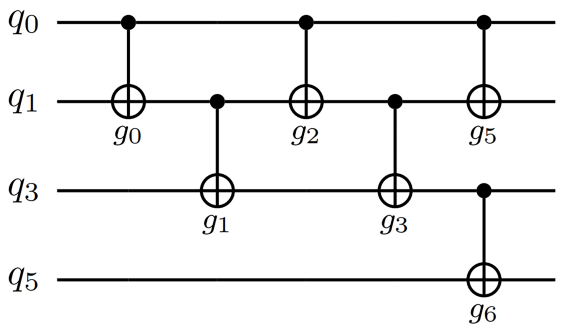

Figure 9. The first part of the quantum circuit.

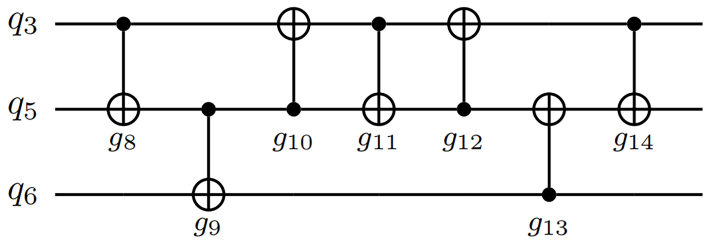

Figure 10. The second part of the quantum circuit.

4.2. Circuit Optimization and Reorganization

After zoning the quantum circuit into local circuits using zoning, a set of local quantum circuits are obtained, which are all composed of nearest neighbor interactions since the selected qubits are arranged linearly. We then use the stencil matching optimization technique for circuit optimization, utilizing a linear template shown in Figures 11 and 12.

The template library used is derived from IBM Qiskit compiler. The non-nearest neighbor template circuits in the library have been modified into linear templates in order to match the topology of the selected qubit.

This paper adopts the template matching algorithm described in [4]. The main idea is as follows: first, all possible initial matches between a gate in the circuit C and a gate in the template T are explored, and the qubits are assigned. Then, while preserving the initial match, the optimal match is maximized. Finally, the matched gates in circuit C are replaced by the inverse of the unmatched gate group in template T.

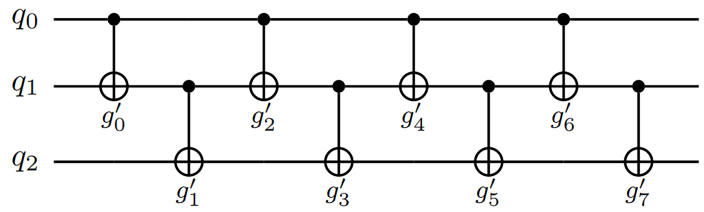

Figure 11. The first template circuit.

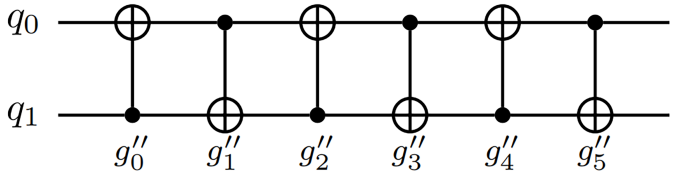

Figure 12. The second template circuit.

As shown in Figure 9, the circuit is first scanned and compared with the template in Figure 11. The first gate g_0 in the circuit matches the first gate g_0' in the template. The qubits q_0, q_1, and q_2 in the template correspond to qubits q_0, q_1, and q_3 in the circuit, respectively.

As the scanning continues, the successor gate g_1 of the gate g_0 in the circuit matches the successor gate g_1' of the gate g_0'. Further scanning reveals that gates g_2, g_3, and g_5 of the circuit match gates g_2', g_3', and g_4' of the template in succession.

However, the successor gate g_6 of gate g_5 in the circuit does not match the successor gate g_5' of gate g_4' in the template, and the matching process is halted. Considering that the number of matched gates is greater than half of the number of gates in the template, replace g_0, g_1, g_2, g_3, and g_5 in the circuit with the remaining gate g_7', g_6', g_5' in the template circuit.

The optimized circuit shown in Figure 13 has two fewer quantum gates than the circuit shown in Figure 9.

Similarly, the quantum circuit in Figure 10 is matched and optimized by the template of Figure 12 to obtain the quantum circuit as shown in Figure 14, reducing two quantum gates.

After block-by-block optimization, the optimized circuit must also be reorganized. The reorganization step involves incorporating an empty quantum circuit within the circuit according to the zoning of the circuit in the fourth part. Then, the quantum gate of each part is included into the circuit according to its qubit information.

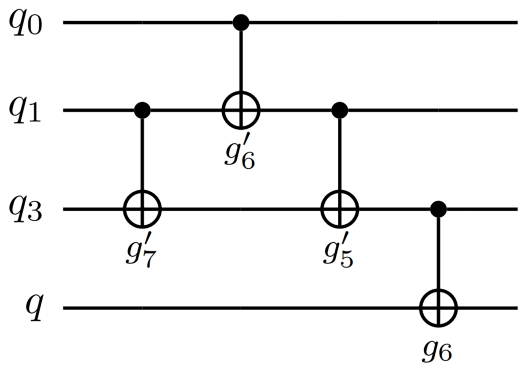

Figure 13. The first part optimizes the post-quantum circuit.

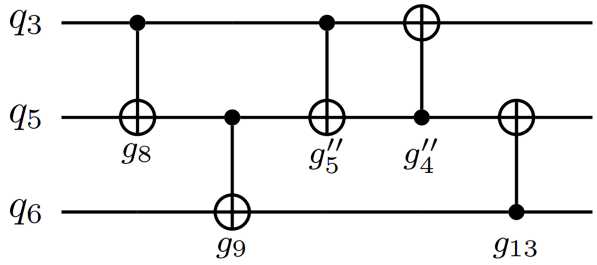

Figure 14. The second part optimizes the post-quantum circuit.

The quantum circuit in Figure 8 is zoned into three parts based on the gate dependency graph. Besides the two optimized local circuits, there are two barrier gates in the middle, as shown in Figure 15. When reorganizing the circuit, the optimized circuit in the first part (Figure 13) is sequentially transferred to an empty quantum circuit according to the qubit and gate order in the original circuit. The two barrier gates are then added accordingly. The quantum gate in the optimized circuit of the second part (Figure 14) is added next. Finally, the resulting circuit shown in Figure 16 is obtained, which reduces the total number of quantum gates by four.

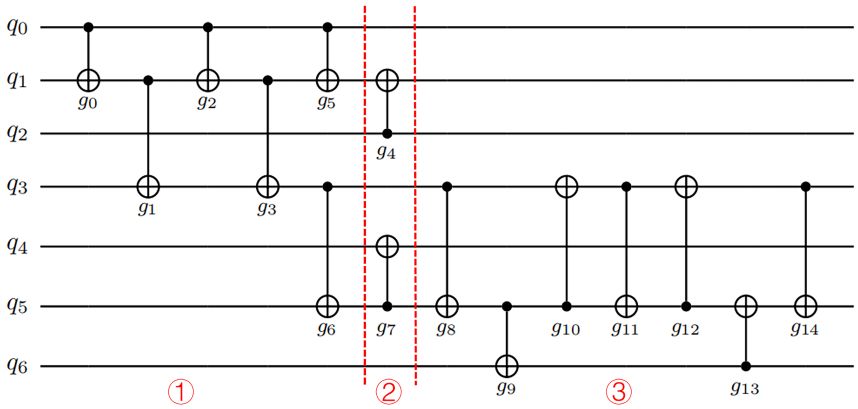

Figure 15. Circuit zoning. The circuit is zoned into three sections, namely 1, 2, and 3, as highlighted in the figure.

This concludes the first round of quantum circuit optimization. To reduce as many quantum gates as possible for the entire quantum circuit, two or more rounds of circuit optimization are required. Compared with the first round of optimization, the overall steps of the subsequent rounds of optimization are unchanged, except for the addition of a selection condition in the linear substructure selection stage. That is, based on the qubits on the linear substructure that have been selected in the $1 \sim i(i > 1)$ round, the selection of the linear substructure in the $i + 1$ round should give priority to the qubits that have not been selected in the previous i round. Until the qubits on the $j(1 \leq j)$ round topology are selected when the linear substructure is selected, then this round of optimization is the last round of optimization of the last round of the current circuit. This ensures that the selection of linear substructures is not repeated for each round and allows each qubit on the topology to be optimized.

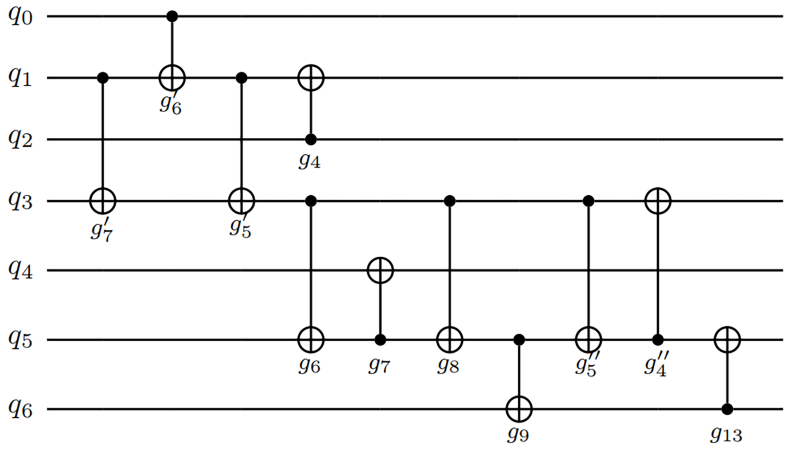

Figure 16. The optimized quantum circuit.

Based on Figure 16, the second round of optimization process builds on the circuit after the first round of optimization. Firstly, the selection of linear substructures prioritizes the unselected qubits, excluding the two qubits already selected Q_2 and Q_4, as shown in Figure 17. Then, since the number of quantum gates between the qubits on the quantum logic circuit corresponding to Q_2Q_1 and Q_4Q_5 is the same, one of the two qubits, Q_2 or Q_4, is randomly chosen as the starting point.

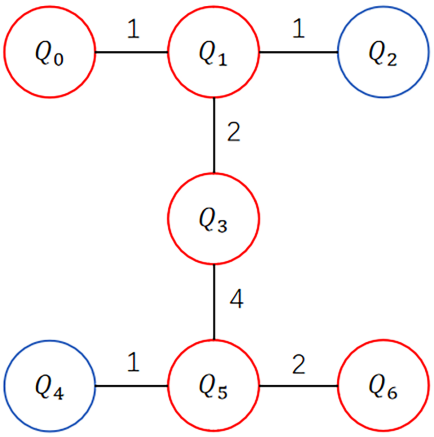

Figure 17. The second round of linear substructure selection. The red circle represents the qubit selected in the first round, while the blue circle represents the newly selected qubit in the second round.

In the selection process, Q_2 is chosen, and Q_0, Q_3 are selected as the successor qubits of Q_1. Because there are more quantum gates between Q_1 and Q_3, Q_3 is selected as the next qubit. Then, Q_5 is selected, and although there are more quantum gates between Q_5 and Q_6, Q_4 is preferred as it is an unselected qubit for the subsequent qubits Q_4 and Q_6 of Q_5.

Therefore, in this way, the second round selects a linear substructure composed of Q_2, Q_1, Q_3, Q_5, Q_4 qubits, covering all the qubits on the entire topology. After the second round of circuit zoning optimization and reorganization, the entire quantum circuit optimization is successfully completed.

As from the reference [27], the linear substructure of a quantum topology can be expressed as an r-noncommutative graph on finite rings. Therefore, Algorithm 3 is suited for determining the optimal path for any given quantum topology that is represented by r-noncommutative graph on finite rings.

The specific other rounds of linear substructure selection algorithms are described as follows:

Algorithm 3: The second round of linear substructure selection algorithm

Input : Topology $G(V, E)$, qubits have been selected
Output: The selected substructure qubits
1. Initialize a list W_1 to store qubits
2. for $v \in V$ do
3. for $e \in E$ do
4. if v_i have the least adjacent connection nodes then
5. for $v_j \in v_i$ connection points do
6. if e_{ij} has maximum weight or v_j not selected then
7. Add $v_i v_j$ to to the list of W_1
8. else if no adjacent nodes can be selected then
9. break
10. end if
11. end for
12. end for
13. return W_1

5. Experimental Results and Analysis

In this experiment, the proposed optimization algorithms were implemented in Python language to verify their feasibility and advantages. To test their efficacy, a set of benchmark circuits, covering multiple qubits with varying quantum architectures, was utilized. The experimental results of this paper were compared with those obtained from the t|ket⟩ compiler and Qiskit optimization in a previous study.

Table 1 employs a linear quantum architecture, while Table 2 employs an ibm_perth quantum architecture. Additionally, Table 3 employs the ibmq_guadalupe quantum architecture for experimentation. The "Circuit name" in the table is the name of the circuit, "n" is the number of qubits of the quantum circuit, "Original CNOTs" is the number of CNOT gates in the initial circuit, "Mapped CNOTs" is the number of CNOT gates optimized and mapped by the compiler after t|ket⟩. "Tket [12]" represents the number of CNOT gates optimized by the t|ket⟩ compiler. "Topt CNOTs" indicates the number of CNOT gates optimized by the method proposed in this article, taking into account the constraints of the quantum architecture connections based on the circuit derived from the "Mapped CNOTs" circuit. "Qiskit [24]" denotes the number of CNOT gates optimized by the Qiskit compiler, and "%" indicates the optimization rate of the proposed method compared to the t | ket⟩ or Qiskit compiler.

Table 1. Experimental comparison results on linear quantum architecture.

Circuit Name	n	Original CNOTs	Mapped CNOTs	Tket [12]	Qiskit [24]	Topt CNots	%	
							With [12]	With [24]
4gt5_75	5	38	63	63	64	38	39.68	40.63
4gt13_90	5	53	64	62	84	45	27.42	46.43
4gt13_91	5	49	64	64	88	48	25.00	45.45
4gt4-v0_78	6	109	180	189	174	131	24.71	30.69
4gt4-v0_79	6	105	163	173	157	113	28.03	34.68
4gt4-v0_80	6	79	151	129	145	131	9.66	−1.55
Average							25.75	32.72

Table 2. Experimental comparison results on ibm_pert.

Circuit Name	*n*	Original CNOTs	Mapped CNOTs	Tket [12]	Qiskit [24]	Topt CNots	%	
							With [12]	With [24]
4gt5_75	5	38	49	47	58	33	29.79	43.10
4gt13_90	5	53	57	57	72	43	24.56	40.28
4gt4-v0_80	6	79	131	127	141	85	33.07	39.75
alu_bdd_288	7	38	75	73	66	60	17.81	9.09
majority_239	7	267	429	423	403	358	15.37	11.17
C17_204	7	205	332	330	332	272	17.58	18.07
ham7_104	7	149	212	210	266	155	26.19	41.73
rd53_131	7	200	339	331	338	296	10.57	12.43
rd53_135	7	134	219	219	248	185	15.53	25.40
Average							21.16	26.78

Table 3. Experimental comparison results on ibmq_guadalupe.

Circuit Name	*n*	Original CNOTs	Mapped CNOTs	Tket [12]	Qiskit [24]	Topt CNots	%	
							With [12]	With [24]
sym9_146	12	148	133	129	211	112	13.18	46.92
cnt3-5_179	16	85	170	170	160	155	8.82	3.13
cnt3-5_180	16	215	361	349	446	321	8.02	28.03
Average							10.01	26.03

The table illustrates the comparison between our proposed method and the t|ket⟩ compiler. In experiments conducted on linear quantum architectures, our method achieves an average reduction of 25.75% in the number of quantum gates through an additional optimization step. For experiments on the ibm_Perth quantum architectures, the average reduction in the number of quantum gates is 21.16%, and for ibmq_guadalupe quantum architectures, it is 10.01%. The maximum optimization effect observed reaches 39.68%, highlighting the effectiveness of our proposed method.

However, it is important to note that as quantum architectures become more complex, the overall optimization effect tends to decrease. This observation is also evident when comparing the results with the Qiskit compiler. This is because choosing a linear substructure introduces more connection constraints between qubits, leading to the circuit being partitioned into more local circuits for optimization.

Nonetheless, the proposed method has shown significant optimization effects on quantum circuits of any scale mapped to any quantum architecture.

6. Conclusions

This paper proposes a new quantum circuit optimization method, namely the quantum circuit template matching optimization method, that takes into account the connectivity constraints of actual quantum topologies. By reducing the number of quantum gates in the circuit, the proposed method successfully satisfies the connectivity constraints and provides improved optimization results.

The experimental results reveal that the method proposed in this paper exhibits a remarkable optimization effect when compared to the t|ket⟩ and Qiskit compiler methods, across various IBM architectures. This method proves to be effective for both one-dimensional and two-dimensional quantum topologies, suggesting its potential to enhance quantum circuit optimization in general.

In future research efforts, we could further improve the optimization effect by leveraging heuristics and applying more intelligent algorithms. Additionally, exploring more types of quantum gates for diverse quantum gate libraries could also benefit the optimization

results. Overall, the proposed method presents a new perspective on quantum circuit optimization with connectivity constraints and exhibits promising potential for wide-ranging applications in the field of quantum computing.

Author Contributions: Conceptualization, X.G. and S.F.; software, X.G.; writing—original draft preparation, X.G.; writing—review and editing, Z.G. and Y.J.; supervision, Z.G.; project administration, Z.G.; funding acquisition, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China under Grant (62072259), in part by Jiangsu Province Natural Science Foundation of China under Grant (BK20151274).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study can be obtained from the authors upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
2. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*, 10th ed.; Cambridge University Press: Cambridge, UK, 2010.
3. Iten, R.; Moyard, R.; Metger, T.; Sutter, D.; Woerner, S. Exact and practical pattern matching for quantum circuit optimization. *ACM Trans. Quantum Comput.* **2022**, *3*, 2643–6808. [CrossRef]
4. Pointing, J.; Padon, O.; Jia, Z.; Ma, H.; Hirth, A.; Palsberg, J. Quanto: Optimizing Quantum Circuits with Automatic Generation of Circuit Identities. *arXiv* **2021**, arXiv:2111.11387.
5. Bandyopadhyay, C.; Wille, R.; Drechsler, R.; Rahaman, H. Post Synthesis-Optimization of Reversible Circuit using Template Matching. In Proceedings of the 24th International Symposium on VLSI Design and Test (VDAT), Bhubaneswar, India, 23–25 July 2020.
6. Bravyi, S.; Shaydulin, R.; Hu, S.; Maslov, D. Clifford Circuit Optimization with Templates and Symbolic Pauli Gates. *Quantum* **2021**, *5*, 580. [CrossRef]
7. Li, G.; Ding, Y.; Xie, Y. 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, Providence, RI, USA, 13–17 April 2019; pp. 1001–1014.
8. Itoko, T.; Raymond, R.; Imamichi, T.; Matsuo, A. Optimization of quantum circuit mapping using gate transformation and commutation. *Integration* **2020**, *70*, 43–50. [CrossRef]
9. Wille, R.; Burgholzer, L.; Zulehner, A. Mapping quantum circuits to IBM QX architectures using the minimal number of SWAP and H operations. In Proceedings of the 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2–6 June 2019; pp. 1–6.
10. Zhu, P.; Zheng, S.; Wei, L. The complexity of quantum circuit mapping with fixed parameters. *Quantum Inf. Process.* **2022**, *21*, 361 . [CrossRef]
11. Zhu, P.; Guan, Z.; Cheng, X. A dynamic look-ahead heuristic for the qubit mapping problem of NISQ computers. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 4721–4735. [CrossRef]
12. Sivarajah, S.; Dilkes, S.; Cowtan, A.; Edgington, W.S.A.; Duncan, R. T|ket⟩: A retargetable compiler for nisq devices. *Quantum Sci. Technol.* **2020**, *6*, 014003. [CrossRef]
13. Steane, A. Quantum computing. *Rep. Prog. Phys.* **1997**, *61*, 117. [CrossRef]
14. Barenco, A.; Bennett, C.H.; Cleve, R.; Divincenzo, D.P.; Margolus, N.; Shor, P. Elementary gates for quantum computation. *Phys. Rev. A* **1995**, *52*, 3457–3467. [CrossRef] [PubMed]
15. Lee, S.; Lee, S.J.; Kim, T. The cost of quantum gate primitives. *J. Mult.-Valued Log. Soft Comput.* **2006**, *12*, 561–573.
16. Kliuchnikov, V.; Maslov, D.; Mosca, M. Asymptotically optimal approximation of single qubit unitaries by Clifford and T circuits using a constant number of ancillary qubits. *Phys. Rev. Lett.* **2013**, *110*, 19. [CrossRef] [PubMed]
17. Nam, Y.; Ross, N.J.; Su, J.; Child, A.M.; Maslow, D. Automated optimization of large quantum circuits with continuous parameters. *NPJ Quantum Inf.* **2018**, *4*, 1. [CrossRef]
18. Patel, T.; Younis, E.; Iancu, C.; Jong, W.D.; Tiwari, D. Robust and Resource-Efficient Quantum Circuit Approximation. *arXiv* **2021**, arXiv:2108.12714.
19. Janzing, D.; Wocjan, D.; Beth, T. "Non-Identity-Check" Is Qma-Complete. *Int. J. Quantum Inf.* **2005**, *3*, 463–473. [CrossRef]
20. Prasad, A.K.; Shende, V.V.; Markov, I.L.; Hayes, J.P.; Patel, K.N. Data structures and algorithms for simplifying reversible circuits. *ACM J. Emerg. Technol. Comput. Syst.* **2006**, *2*, 277–293. [CrossRef]

21. Maslov, D.; Dueck, G.W.; Miller, D.M.; Negrevergne, C. Quantum circuit simplification and level compaction. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2008**, *27*, 436–444. [CrossRef]
22. Duncan, R.; Kissinger, A.; Pedrix, S.; Wetering, J. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *Quantum* **2020**, *4*, 279. [CrossRef]
23. IBM Quantum Experience. Available online: https://quantum-computing.ibm.com/ (accessed on 11 December 2022).
24. Qiskit—An Open-Source Framework for Working with Noisy Quantum Computers at the Level of Pulses, Circuits, and Algorithms. Available online: https://github.com/Qiskit (accessed on 14 May 2023).
25. Rahman, M.M.; Dueck, G.W. Properties of quantum templates. In Proceedings of the 4th International Workshop, RC 2012, Berlin/Heidelberg, Germany, 2–3 July 2012; pp. 125–137.
26. Rahman, M.M.; Dueck, G.W.; Horton, J.D. *An Algorithm for Quantum Template Matching*; Association for Computing Machinery: New York, NY, USA, 2015; Volume 11, pp. 1550–4832.
27. Nath, R.K.; Sharma, M.; Dutta, P.; Shang, Y. On r-Noncommuting Graph of Finite Rings. *Axioms* **2021**, *10*, 233. [CrossRef]