entropy

IMPACT FACTOR 2.738

Indexed in: PubMed

Article

A Method Based on Timing Weight Priority and Distance Optimization for Quantum Circuit Transformation

Yang Qian, Zhijin Guan, Shenggen Zheng and Shiguang Feng

Special Issue
Quantum Machine Learning 2022

Edited by
Prof. Dr. Andreas Wichert

MDPI

https://doi.org/10.3390/e25030465

entropy

MDPI

Article

A Method Based on Timing Weight Priority and Distance Optimization for Quantum Circuit Transformation

Yang Qian [1], Zhijin Guan [1,*], Shenggen Zheng [2,*] and Shiguang Feng [1,3,*]

[1] School of Information Science and Technology, Nantong University, Nantong 226000, China
[2] Peng Cheng Laboratory, Shenzhen 518055, China
[3] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China
* Correspondence: guan.zj@ntu.edu.cn (Z.G.); zhengshg@pcl.ac.cn (S.Z.); shigfeng@ntu.edu.cn (S.F.)

Abstract: In order to implement a quantum circuit on an NISQ device, it must be transformed into a functionally equivalent circuit that satisfies the device's connectivity constraints. However, NISQ devices are inherently noisy, and minimizing the number of SWAP gates added to the circuit is crucial for reducing computation errors. To achieve this, we propose a subgraph isomorphism algorithm based on the timing weight priority of quantum gates, which provides a better initial mapping for a specific two-dimensional quantum architecture. Additionally, we introduce a heuristic swap sequence selection optimization algorithm that uses a distance optimization measurement function to select the ideal sequence and reduce the number of SWAP gates, thereby optimizing the circuit transformation. Our experiments demonstrate that our proposed algorithm is effective for most benchmark quantum circuits, with a maximum optimization rate of up to 43.51% and an average optimization rate of 13.51%, outperforming existing related methods.

Keywords: quantum circuit transformation; qubit mapping; subgraph isomorphism; heuristic optimization

Citation: Qian, Y.; Guan, Z.; Zheng, S.; Feng, S. A Method Based on Timing Weight Priority and Distance Optimization for Quantum Circuit Transformation. *Entropy* **2023**, *25*, 465. https://doi.org/10.3390/e25030465

Academic Editor: Andreas Wichert

Received: 6 February 2023
Revised: 2 March 2023
Accepted: 6 March 2023
Published: 7 March 2023

1. Introduction

Quantum computing, a new computing paradigm that leverages the superposition and entanglement features of quantum mechanics, has the potential to solve many problems faster than classical computers, such as integer factorization [1] and linear equations [2]. Currently, quantum computing has entered the era of noisy intermediate-scale quantum (NISQ) devices [3]. In contrast to the idealized quantum circuit model, physical quantum architectures have a connectivity constraint, limiting the set of allowable two-qubit gates between specific pairs of qubits. To ensure that a quantum circuit is functionally equivalent to the desired computation and satisfies the connectivity constraint, quantum circuit transformations must be applied. However, NISQ devices suffer from crosstalk noise, which results from unexpected interactions or uncorrected control of qubits. This noise compromises the fidelity of the final circuit execution results [4]. Therefore, minimizing the number of SWAP gates added to the circuit during the transformation process is a critical task for circuit optimization.

The process of transforming a quantum circuit can be divided into two sub-procedures. The first involves finding an initial mapping, which maps logical qubits to physical qubits. The second sub-procedure involves handling quantum gates that violate the interaction constraint in the current mapping by inserting SWAP gates to lead to neighboring qubits. However, determining the minimal number of SWAP gates required for the quantum circuit transformation is an NP complete problem [5]. To solve this problem, various methods have been proposed [6–15]. One type of method formulates the problem mathematically and uses solvers such as integer linear programming [16], satisfiability module theory [4], constraint planning [17], or Boolean solvers [18] to find solutions. These methods are effective for

small quantum circuits. The second type of method uses heuristic algorithms to solve the problem. For example, the backward traversal-based mapping method SABRE [6], Monte Carlo tree search [8], A* search algorithm [7,19], good initial mapping generation [9], Bridge gate insertion [10], reversibility-based comparison of forward and reverse circuit transformation processes [11], greedy algorithm [12], simulated annealing [13], subgraph isomorphism-based mapping [14], and an algorithm based on dynamic look-ahead heuristic cost functions [15] have been proposed. These heuristic algorithms use different evaluation functions to determine the best mapping strategy, based on various factors such as the number of two-qubit quantum gates, the depth of the circuit, the distance between the control qubit and the target qubit, or the topology of the quantum circuit. Some methods also consider the dependency of quantum gates in the circuit and prioritize the execution of preceding gates.

In this paper, we follow a similar approach to the literature [14] by using the number of additional CNOT gates as a metric for measurement, and we utilize subgraph isomorphism as the basic algorithm in the initial mapping. However, in addition to this, we consider the execution order of the quantum gates in the logical quantum circuit during the initial mapping process. Furthermore, we propose a forward-looking heuristic algorithm during the routing process to compare the quantum gate interaction distances, particularly in cases where the interactability rates are equal, to find a better routing path. The proposed method achieves the final transformation of logical quantum circuits to executable physical quantum circuits.

The process of transforming a quantum circuit can be divided into two main steps: (i) initial qubit mapping and (ii) insertion of SWAP gates. In the initial qubit mapping step, we first assign a timing weight to each edge in the circuit's interaction graph and then use a subgraph isomorphism algorithm based on these timing weights to find an initial mapping between the logical qubits and physical qubits. In the SWAP gates insertion step, there may be several possible sequences of SWAP gates that can be applied. To determine the best sequence, we define an interactivity value and a distance optimization measure for each sequence. We propose a sequence selection optimization algorithm based on maximizing the distance measure optimization function, which selects the SWAP gates sequence with the highest value. The experimental results demonstrate the effectiveness of our algorithm.

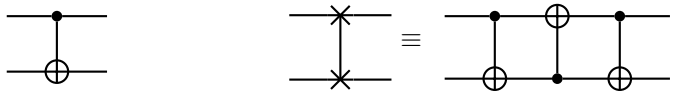
The paper is structured as follows. In Section 2, we provide definitions and notations related to quantum gates and circuits. Section 3 presents our initial mapping algorithm based on timing weight subgraph isomorphism. In Section 4, we propose a SWAP gate sequence selection optimization algorithm for quantum circuit transformation. In Section 5, we evaluate our approach on benchmark quantum circuits and compare it with a state-of-the-art method. We conclude the paper in Section 6.

2. Preliminaries

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

2.1. Quantum Gate and Quantum Circuit

In quantum computing, quantum bits (qubits) are the fundamental unit of quantum information [20]. In contrast to classical bits, which only have two states (0 and 1), the state $|\phi\rangle$ of a qubit is a superposition of the two states $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. Quantum gates are used to operate on qubits in quantum computers. Single-qubit gates operate on a single qubit, while two-qubit gates operate on two qubits. A two-qubit gate g is denoted as $g = \langle p, q\rangle$ and indicates that g operates on qubits p and q. The CNOT gate (see Figure 1a) is an example of a two-qubit gate, where p is the control qubit and q is the target qubit. If p is in state 1, the CNOT gate flips the state of q, and if p is in state 0, q remains unchanged. A SWAP gate $\langle p, q\rangle$ exchanges the states of p and q, and can be realized by cascading three CNOT gates (see Figure 1b).



(**a**) (**b**)

Figure 1. The CNOT gate and the SWAP gate. (**a**) A CNOT gate. (**b**) A SWAP gate and its realization by CNOT gates.

Quantum circuits are used to describe quantum algorithms and are composed of qubits, quantum gates, measurement gates, classical registers, and so on. We use $LC = (Q, C)$ to denote a quantum circuit, where Q and C represent the set of qubits and the set of quantum gates in the circuit, respectively. Since the CNOT gate and all single-qubit gates are widely used as universal quantum gates set, and the single-qubit gates are compliant with the connectivity constraint, we only consider quantum circuits composed of two-qubit gates in this paper.

Example 1. *Figure 2 depicts a quantum circuit* $LC = (Q, C)$*, where* $Q = \{q_0, q_1, q_2, q_3\}$ *and* $C = \{g_0 = \langle q_2, q_0\rangle, g_1 = \langle q_3, q_2\rangle, g_2 = \langle q_0, q_3\rangle, g_3 = \langle q_0, q_2\rangle, g_4 = \langle q_3, q_2\rangle, g_5 = \langle q_0, q_3\rangle, g_6 = \langle q_3, q_1\rangle, g_7 = \langle q_0, q_1\rangle\}$*, consisting of four qubits and eight CNOT gates.*

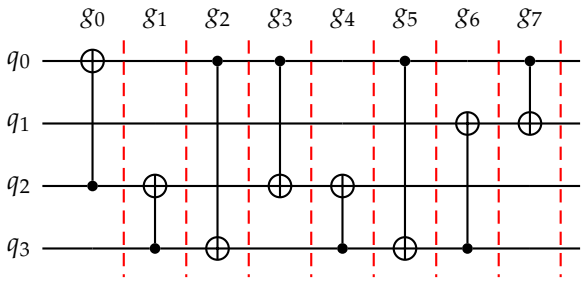
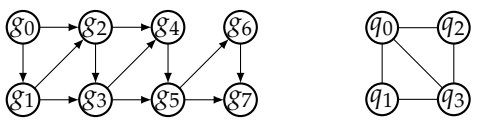


Figure 2. A quantum circuit with four qubits and eight CNOT gates.

2.2. Dependency Graph and Interaction Graph

Given a quantum circuit $LC = (Q, C)$, we define two graphs: the dependency graph, DG, and the interaction graph, IG. DG is a directed acyclic graph whose nodes are the gates in C. There is a directed edge from gate g_i to gate g_j in DG if g_i operates on a qubit q and g_j is the next gate that operates on q after g_i. IG is an undirected graph whose nodes are the qubits in Q. There is an edge between two qubits q_i and q_j in IG if they are operated by some gate. Figure 3 shows the dependency graph and interaction graph of the quantum circuit in Figure 2. The dependency graph captures the dependencies between gates in the circuit, while the interaction graph shows the connectivity of qubits in the circuit.



(**a**) (**b**)

Figure 3. The dependency graph and interaction graph of the quantum circuit in Figure 2. The dependency graph, shown in (**a**), is a directed acyclic graph with nodes representing gates in the circuit and directed edges representing dependencies between gates. The interaction graph, shown in (**b**), is an undirected graph with nodes representing qubits in the circuit and edges representing interactions between qubits.

The dependency graph is a useful tool to represent the execution order of quantum gates in a quantum circuit, while the interaction graph is commonly used to represent the interaction relationships between qubits in a quantum circuit [14].

3. Initial Qubit Mapping

A physical quantum device can be represented by its coupling graph CG, which is an undirected graph (V, E) where each qubit in the device is a node in V, and there is an edge $(q_i, q_j) \in E$ between two nodes q_i and q_j if they can be operated by a two-qubit gate in the device. For instance, Figure 4 shows the coupling graph of IBM QX20 Tokyo.

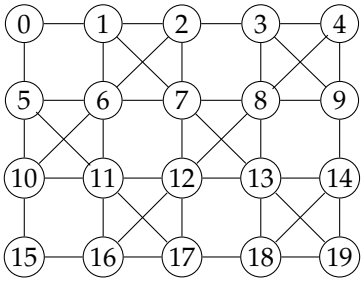


Figure 4. The coupling graph of IBM QX20 Tokyo.

Given a quantum circuit $LC = (Q, C)$ and a coupling graph $CG = (V, E)$, an initial mapping π is an injection from Q to V. To find an effective initial mapping, it is important to consider the following two conditions when designing the algorithm:

1. Logical qubits that interact frequently should be mapped to physically adjacent qubits.
2. Qubits with high execution priority, based on the order of quantum gates in the circuit, should be mapped to adjacent physical qubits.

By taking into account these conditions, we can design an initial mapping algorithm that can improve the performance and efficiency of quantum circuit execution on physical devices.

3.1. Timing Weight

To further improve the effectiveness of the initial mapping algorithm, we can introduce another metric called timing weight. The timing weight takes into account the execution order of quantum gates and the interaction relationships between qubits in the quantum circuit. In this subsection, we define the timing weight for quantum gates and edges in the interaction graph and explain how it can be used to design an improved initial mapping algorithm.

Definition 1. *Given a quantum circuit* $LC = (Q, C)$, *where* $Q = \{q_0, q_1, \ldots, q_m\}$ *and* $C = \{g_0, \ldots, g_{n-1}\}$, *the timing weight,* k_i, *for each quantum gate* g_i *is*

$$k_i = n - i, \tag{1}$$

where $0 \leq i \leq n-1$ *is the execution order of* g_i.

Definition 2. *Let IG be the interaction graph of LC. For each edge* (p, q) *in IG, where* p, q *are qubits in LC, the timing weight,* $\omega_{(p,q)}$, *of* (p, q) *is*

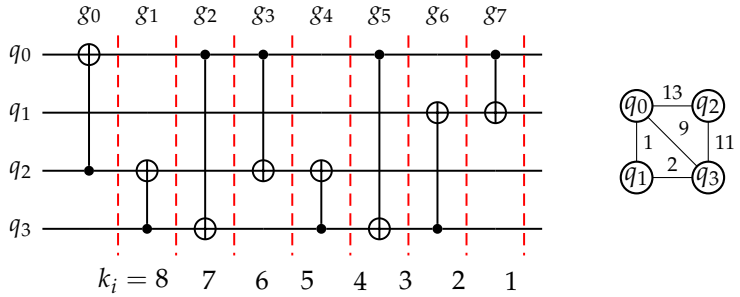
$$\omega_{(p,q)} = \sum_{g_i = \langle p,q \rangle} k_i + \sum_{g_j = \langle q,p \rangle} k_j, \tag{2}$$

where $k_i(k_j)$ *is the timing weight of* $g_i(g_j)$.

The timing weight measures the importance of a quantum gate in the execution order of a quantum circuit, with lower weights assigned to gates executed later. The timing

weight of an edge in the interaction graph represents the total timing weight of the gates that interact through that edge.

Example 2. *Figure 5 displays the timing weights of the gates in the circuit of Figure 2 and the timing weights of the edges in its interaction graph. The edge* (q_0, q_2) *has the largest weight, indicating that the gates associated with these qubits are executed first in the circuit. As* q_0 *and* q_2 *contain the most CNOT gates in the circuit, they should be mapped as nearest neighbors.*



(**a**) (**b**)

Figure 5. The timing weights of gates and edges. (**a**) The timing weight of each gate in the circuit of Figure 2. (**b**) The timing weight of each edge in the interaction graph of the circuit.

3.2. Initial Mapping Based on Timing Weight

To construct an initial mapping for a quantum circuit *LC* and a coupling graph *CG*, we first compute the interaction graph *IG* of *LC* with timing weights assigned to each edge. If *IG* is isomorphic to a subgraph of *CG*, we can construct an initial mapping from the isomorphism without adding any SWAP gates. However, when *IG* cannot be directly mapped to *CG*, we use a partial subgraph isomorphism based on timing weight. This means that edges with larger timing weights are given higher priority in the mapping process.

The timing weight of each edge in *IG* not only indicates the qubits with more CNOT executions but also the execution order of the CNOT gates in the circuit. Thus, it is an important factor in finding an optimal initial mapping that provides a better precondition for subsequent quantum circuit transformations. We propose an initial mapping algorithm based on timing weight subgraph isomorphism, as outlined in Algorithm 1. The algorithm takes *LC* and *CG* as inputs and constructs an initial mapping by searching every edge from the one with the highest timing weight.

Algorithm 1 outlines the steps for constructing an initial mapping based on timing weight subgraph isomorphism. To begin, we initialize the set of edges $Edge_set$ of *IG*. Lines 2–8 compute the interaction graph, *IG*, of *LC* with timing weights for all edges. Line 9 sorts the edges from the largest timing weight to the smallest timing weight. Lines 10–16 construct a graph *ig* starting from the edge with the largest timing weight and add the edge to *ig* if there is a subgraph isomorphism from *ig* to *CG*. If there is no subgraph isomorphism, the edge is skipped. This process is repeated until all the edges in $Edge_set$ are traversed. Finally, an initial mapping is constructed from *ig* and saved in *results*. It is important to note that the initial mapping obtained may not be unique, as it depends on the subgraph isomorphism found from *ig* to *CG* in the algorithm.

Algorithm 1 Initial mapping based on timing weight subgraph isomorphism

Input: A quantum circuit $LC = (Q, C)$ and a coupling graph $CG = (V, E)$.
Output: An initial mapping from Q to V.

Initialize: $results \leftarrow \emptyset$; $Edge\_set \leftarrow \emptyset$
for each gate $g_i \in LC$ **do**
  $k_i \leftarrow |C| - i$
  if $g_i = \langle p, q \rangle$ or $g_i = \langle p, q \rangle$ **then**
    $\omega_{(p,q)} \leftarrow sum(k_i)$
    $Edge\_set \leftarrow [(p, q), \omega_{(p,q)}]$
  end if
end for
$Edge\_set.sort()$
while $Edge\_set \neq \emptyset$ **do**
  if $ig \cup Edge\_set.pop()$ is isomorphic to a subgraph of CG **then**
    $ig \leftarrow ig \cup Edge\_set.pop()$
  else
    $skip(Edge\_set.pop())$
  end if
end while
$results \leftarrow mapping(ig, CG)$
return $results$

Example 3. *By applying Algorithm 1 to the quantum circuit shown in Figure 2 and the coupling graph of IBM QX20 shown in Figure 4, we can obtain three initial mappings (colored in red) as shown in Figure 6. It is important to note that the initial mapping obtained may not be unique, as it depends on the subgraph isomorphism found from the algorithm.*

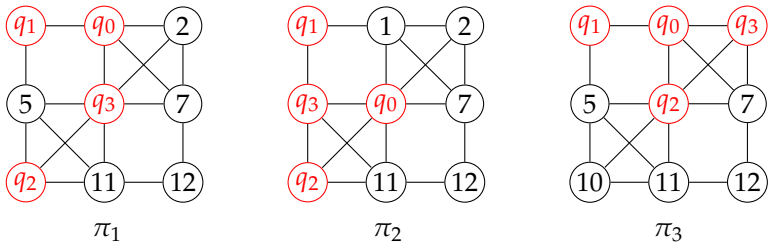


Figure 6. Three initial mappings from the circuit in Figure 2 to IBM QX20.

4. Quantum Circuit Transformation and Optimization

Given the quantum circuit LC, the coupling graph CG, and the mapping π from LC to CG, we need to ensure that every quantum gate in LC is mapped to adjacent qubits in CG. If a gate violates this interaction condition, we can add SWAP gates to the circuit to make the two qubits operated by the gate adjacent. This process can be repeated for every edge in CG. We use $SWAP(e)$ to denote the application of a SWAP gate on the two ends of e. After performing a swap operation, a new mapping π' can be obtained by updating π. For instance, in Figure 6, the initial mapping π_1 is $\{q_0 \rightarrow 1, q_1 \rightarrow 0, q_2 \rightarrow 10, q_3 \rightarrow 6\}$. We can apply $SWAP(5, 6)$ and $SWAP(1, 6)$ to obtain the mapping $\pi_2 = \{q_0 \rightarrow 6, q_1 \rightarrow 0, q_2 \rightarrow 10, q_3 \rightarrow 5\}$.

4.1. Swap Sequence Selection

During the quantum circuit transformation process, a sequence of SWAP gates is added to ensure that every gate in the circuit satisfies the interaction condition with respect to the coupling graph. However, this sequence is not unique, and we aim to find the shortest sequence that minimizes the number of gates in the resulting circuit.

Let π be a mapping from the quantum circuit LC to the coupling graph CG, and let

$$a = (SWAP(e_1), SWAP(e_2), \ldots, SWAP(e_n))$$

be a sequence of SWAP gates, where each $e_i (1 \leq i \leq n)$ is an edge of CG. We use g_{ex} to denote the number of gates in LC that are executable after applying the sequence a from mapping π. To measure the effectiveness of the transformation, we define the interactivity value:

$$G_{val}(\pi, a) = \frac{g_{ex}}{len(a)}, \tag{3}$$

where $len(a) = n$ is the number of SWAP gates in a. A larger value of $G_{val}(\pi, a)$ indicates that the sequence a is more effective in transforming the circuit. Therefore, we aim to find the sequence b that maximizes $G_{val}(\pi, b)$ for a given mapping π.

Now let us consider an example to see how the swap sequence selection algorithm works in practice.

Example 4. *Consider the quantum circuit shown in Figure 2 and the initial mapping* $\pi_1 = q_0 \to 1, q_1 \to 0, q_2 \to 10, q_3 \to 6$ *in Figure 6. The first quantum gate that violates the interactive constraint is* $g_0 = \langle q_2, q_0 \rangle$*. To address this issue, we explore four different swap ways and the number of gates that can be executed for each one, as follows:*

1. $SWAP(5,6), SWAP(1,6)$*: seven gates can be executed;*
2. $SWAP(2,6), SWAP(10,6)$*: seven gates can be executed;*
3. $SWAP(1,7), SWAP(10,6)$*: four gates can be executed;*
4. $SWAP(0,1), SWAP(5,0)$*: six gates can be executed.*

We can then evaluate the validity of each sequence using the metric G_{val}*, and obtain* $G_{val}(\pi_1, 1) = 7/2$*,* $G_{val}(\pi_1, 2) = 7/2$*,* $G_{val}(\pi_1, 3) = 2$*, and* $G_{val}(\pi_1, 4) = 3$*. Based on these values, we conclude that the candidate sequences are (i) and (ii).*

4.2. Sequence Selection Optimization

For large-scale quantum circuits, there are often multiple swap sequences that satisfy the interactivity constraint, as shown in Example 4. In this subsection, we propose a sequence selection optimization algorithm to find a better swap sequence from the candidate sequences.

Given the quantum circuit LC, the coupling graph CG, and the mapping π from LC to CG, we use $dist_{ph}(g, \pi)$ to denote the length of the shortest path between $\pi(p)$ and $\pi(q)$ in CG for every gate $g = \langle p, q \rangle$ in LC. Let a be a sequence of SWAP gates, and let $\pi[a]$ denote the new mapping obtained by applying a to π. We define the set $S = \{g \mid dist_{ph}(g, \pi[a]) < dist_{ph}(g, \pi)\}$, which contains the gates whose execution order is affected by the sequence a. We then define the distance optimization measure as follows:

$$\Omega_g(\pi, a) = \sum_{g_i \in S} k_i, \tag{4}$$

where k_i is the timing weight of g_i.

The value of $\Omega_g(\pi, a)$ depends on the timing weight of every gate in S. A gate g_i with a large value of k_i means that g_i has a high execution order. Therefore, $\Omega_g(\pi, a)$ indicates the priority of the sequence a; a sequence with a larger $\Omega_g(\pi, a)$ should be selected first. To select the best swap sequence, we compute the $\Omega_g(\pi, a)$ values for each candidate sequence using Equation (4), and choose the sequence with the largest value. The swap sequence selection optimization algorithm is presented in Algorithm 2. The inputs of the algorithm are a quantum circuit, a coupling graph, and an initial mapping. The output is a transformed quantum circuit that is compliant with the connectivity constraint.

Algorithm 2 Sequence selection optimization algorithm

Input: A quantum circuit LC, a coupling graph CG, and an initial mapping π_0.
Output: A quantum circuit PC that is compliant with connectivity constraint of CG.

1: **Initialize:** $swap \leftarrow \emptyset$; $a \leftarrow \emptyset$; $\pi \leftarrow \pi_0$; $PC \leftarrow \emptyset$
2: **for** each gate $g = \langle p, q \rangle \in LC$ **do**
3: **if** $dist_{ph}(g, \pi) = 1$ **then**
4: $LC \leftarrow LC.remove(g)$
5: $PC \leftarrow PC.add(g)$
6: **else**
7: $\max G_{val} \leftarrow G_{val}(\pi, (a, SWAP)).sort()$
8: (compute the G_{val} for all possible SWAP gates and save the largest ones)
9: **for** each $SWAP \in \max G_{val}$ **do**
10: $swap \leftarrow \max \Omega_g(\pi, (a, SWAP))$
11: (compute the Ω_g for all sequences in $\max G_{val}$ and take one of the largest)
12: **end for**
13: $a \leftarrow (a, swap)$
14: $\pi \leftarrow \pi[swap]$
15: $PC \leftarrow PC.add(sawp)$
16: $PC \leftarrow PC.add(g)$
17: $LC \leftarrow LC.remove(g)$
18: **end if**
19: **end for**
20: **return** PC

The algorithm starts by initializing the necessary variables. It adds all quantum gates that meet the connectivity constraint in PC and removes them from LC according to their execution order in the current mapping, as described in Lines 3–5. If no gate can be added to PC, indicating that SWAP gates must be inserted, the algorithm computes the G_{val} for all possible SWAP gate sequences and saves those with the largest value in $\max G_{val}$, as shown in Line 7. Next, Lines 9–12 compute the Ω_g for all sequences in $\max G_{val}$ and selects the one that has the largest value. Finally, Lines 13–17 update LC, PC, π, and a based on the SWAP gates added.

To illustrate how the sequence selection optimization algorithm works, let us consider the following example.

Example 5. *In Example 4, we compare the two SWAP gate sequences (i) and (ii), which have the same G_{val} value. Using Equation (4), we obtain the following distances and cost values:*

$$\begin{aligned}
\pi_1: \quad & dist_{ph}(g_0, \pi_1) = dist_{ph}(g_3, \pi_1) = dist_{ph}(g_6, \pi_1) = 2 \\
& dist_{ph}(g_1, \pi_1) = dist_{ph}(g_2, \pi_1) = dist_{ph}(g_4, \pi_1) = 1 \\
& dist_{ph}(g_5, \pi_1) = dist_{ph}(g_7, \pi_1) = 1 \\
\pi_1[(i)]: \quad & dist_{ph}(g_i, \pi_1[(i)]) = 1\ (0 \leq i \leq 6), dist_{ph}(g_7, \pi_1[(i)]) = 2 \\
& \Omega_g(\pi_1, (i)) = 8 + 5 + 2 = 15 \\
\pi_1[(ii)]: \quad & dist_{ph}(g_i, \pi_1[(ii)]) = 1\ (0 \leq i \leq 5), dist_{ph}(g_6, \pi_1[(ii)]) = 2, \\
& dist_{ph}(g_7, \pi_1[(ii)]) = 1 \\
& \Omega_g(\pi_1, (ii)) = 8 + 5 = 13
\end{aligned}$$

Since both sequences have the same G_{val} value, we choose the sequence with the higher values according Algorithm 2. Therefore, we choose sequence (i) as the optimal swap sequence.

5. Evaluation

To assess the effectiveness of the proposed methods, we now turn to evaluate their performance on a variety of quantum circuits. We provide experimental results and comparisons with existing approaches in this section.

The proposed methods were implemented in Python, and the experiments were conducted on a Windows 10 machine with an Intel Core i7 processor and 16 GB of RAM. The quantum circuits used in the experiments were publicly available benchmarks evaluated in [14]. The experimental comparison baseline is the number of CNOT gates inserted by the quantum circuit transformation. In this paper, we classify quantum circuits with less than 1000 gates as small scale, those with greater than 1000 and less than 10,000 gates as medium scale, and those with more than 10,000 gates as large scale.

Tables 1–3 show the results of the experiments for small scale, medium scale, and large scale circuits, respectively. The first column in each table lists the benchmark, the second column "QubitNu" shows the number of qubits in the circuit, and the third column "GateNu" shows the number of CNOT gates in the circuit. The fourth and fifth columns, labeled "AuxGt", show the number of auxiliary CNOT gates added by the transformation algorithm in [6,14], respectively, and serve as the main comparative baseline for the experiments. The sixth column shows the number of auxiliary CNOT gates added by the algorithm proposed in this paper. The seventh and eighth columns, labeled "Comp.", show the optimization rate compared with the literature [6,14]. The last row of each table shows the average optimization rate.

In order to evaluate the effectiveness of the proposed quantum circuit transformation optimization algorithm, we compare the experimental results with two previous algorithms, Sabre [6] and Topgraph [14]. Sabre is integrated in the IBMQ quantum cloud platform, which represents the current optimal level of IBMQ, and Topgraph is the basis of algorithm improvement in this paper. Therefore, they are selected as the baselines for the experiments in this paper. From the experimental results in the three tables, it is evident that the quantum circuit transformation optimization algorithm in this paper outperforms Sabre, with the highest average optimization rate reaching 70.15%. In comparison with Topgraph, the algorithm proposed in this paper achieves a minimum average optimization rate of 4.30% on small-scale circuits and 21.79% on large-scale circuits. Although two quantum circuits in the medium-scale circuit are not optimized, the average optimization rate still reaches 22.04%. Overall, the proposed algorithm has a significant optimization effect on circuits of any scale, with the larger circuits showing more obvious optimization.

Table 1. The experimental results on small-scale circuits.

Benchmark			AuxGt in [6]	AuxGt in [14]	AuxGt	Comp.	
Circuit	**QubitNu**	**GateNu**				**With [6]**	**With [14]**
4mod5_v1_22	5	21	0	0	0	0.00%	0.00%
mod5mils_65	5	35	0	0	0	0.00%	0.00%
decod24v2_43	4	52	0	0	0	0.00%	0.00%
4gt13_92	5	66	0	0	0	0.00%	0.00%
qft_10	10	200	54	39	33	38.89%	15.38%
qft_16	16	512	186	153	117	37.10%	25.53%
ising_model_10	10	480	0	0	0	0.00%	0.00%
ising_model_13	13	633	0	0	0	0.00%	0.00%
ising_model_16	16	786	0	0	0	0.00%	0.00%
rd84_142	15	343	105	72	66	37.14%	8.33%
Average						11.31%	4.72%

Table 2. The experimental results on medium-scale circuits.

Benchmark			AuxGt in [6]	AuxGt in [14]	AuxGt	Comp.	
Circuit	**QubitNu**	**GateNu**				**With [6]**	**With [14]**
sym6_145	7	3888	1272	513	405	68.16%	21.05%
z4_268	11	3073	1365	630	363	73.41%	42.38%
radd_250	13	3213	1275	555	576	54.82%	−3.78%
cycle10_2_110	12	6050	2622	1194	969	63.04%	18.84%
adr4_197	13	3439	1614	630	648	59.85%	−2.86%
misex1_241	15	4813	1521	786	444	70.81%	43.51%
rd73_252	10	5321	2133	1095	732	65.68%	33.15%
square_root_7	15	7630	2598	1338	1017	60.85%	23.99%
Average						64.58%	22.04%

Table 3. The experimental results on large-scale circuits.

Benchmark			AuxGt in [6]	AuxGt in [14]	AuxGt	Comp.	
Circuit	**QubitNu**	**GateNu**				**With [6]**	**With [14]**
co14_215	15	17,936	8982	4257	2658	70.41%	37.56%
rd84_253	12	13,658	6147	2352	1827	70.28%	22.32%
sqn_258	10	10,223	4344	1578	1212	72.10%	23.19%
sym9_193	11	34,881	16,653	5589	5361	67.81%	4.08%
Average						70.15%	21.79%

6. Conclusions

In this paper, we proposed a timing-based optimization algorithm for quantum circuit transformation that defines a timing weight for every edge in the interaction graph of the circuit. By using a subgraph isomorphism algorithm based on timing weights, we can obtain a high-quality initial mapping. We also proposed a sequence selection optimization algorithm based on the distance measure optimization function for selecting the best SWAP gate sequence. The experimental results show that our algorithm outperforms the existing benchmarks for arbitrary scale quantum circuits, demonstrating its effectiveness. However, there are other factors that can affect the execution of quantum computation, such as quantum gate execution errors and qubit coherence time, which could be incorporated into a study for comprehensive consideration. Our future research direction will be to adapt the algorithm in this paper to account for these factors.

Author Contributions: Conceptualization, Y.Q. and S.Z.; software, Y.Q.; writing—original draft preparation, Y.Q.; writing—review and editing, S.Z. and S.F.; 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 research was funded in part by the National Natural Science Foundation of China under grant 62072259 and by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under grant number SJCX22_1617. Shenggen Zheng was supported by the Major Key Project of PCL.

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. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
2. Harrow, A.W.; Hassidim, A.; Lloyd, S. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.* **2009**, *103*, 150502. [CrossRef] [PubMed]
3. Preskill, J. Quantum Computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
4. Murali, P.; Baker, J.M.; Javadi-Abhari, A.; Chong, F.T.; Martonosi, M. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, Providence, RI, USA, 13–17 April 2019; pp. 1015–1029.
5. Zhu, P.; Zheng, S.; Wei, L.; Cheng, X.; Guan, Z.; Feng, S. The complexity of quantum circuit mapping with fixed parameters. *Quantum Inf. Process.* **2022**, *21*, 361. [CrossRef]
6. 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.
7. Zulehner, A.; Gasser, S.; Wille, R. Exact Global Reordering for Nearest Neighbor Quantum Circuits Using A. In Proceedings of the Reversible Computation, Kolkata, India, 6–7 July 2017; Phillips, I., Rahaman, H., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 185–201.
8. Zhou, X.; Feng, Y.; Li, S. A Monte Carlo Tree Search Framework for Quantum Circuit Transformation. In Proceedings of the 39th International Conference on Computer-Aided Design, Virtual, 2–5 November 2020; pp. 1–7.
9. Chhangte, L.; Chakrabarty, A. Mapping Quantum Circuits in IBM Q Devices Using Progressive Qubit Assignment for Global Ordering. *New Gener. Comput.* **2022**, *40*, 311–338. [CrossRef]
10. Niu, S.; Suau, A.; Staffelbach, G.; Todri-Sanial, A. A Hardware-Aware Heuristic for the Qubit Mapping Problem in the NISQ Era. *IEEE Trans. Quantum Eng.* **2020**, *1*, 3101614. [CrossRef]
11. Zhu, P.; Feng, S.; Guan, Z. An Iterated Local Search Methodology for the Qubit Mapping Problem. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2021**, *41*, 2587–2597. [CrossRef]
12. Childs, A.M.; Schoute, E.; Unsal, C.M. Circuit Transformations for Quantum Architectures. In Proceedings of the 14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019), College Park, MD, USA, 3–5 June 2019; van Dam, W., Mancinska, L., Eds.; Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik: Dagstuhl, Germany, 2019; Volume 135, pp. 3:1–3:24.
13. Zhou, X.; Li, S.; Feng, Y. Quantum circuit transformation based on simulated annealing and heuristic search. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 4683–4694. [CrossRef]
14. Li, S.; Zhou, X.; Feng, Y. Qubit mapping based on subgraph isomorphism and filtered depth-limited search. *IEEE Trans. Comput.* **2020**, *70*, 1777–1788. [CrossRef]
15. 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]
16. Bhattacharjee, D.; Chattopadhyay, A. Depth-Optimal Quantum Circuit Placement for Arbitrary Topologies. *arXiv* **2017**, arXiv:1703.08540.
17. Booth, K.; do, M.; Beck, J.; Rieffel, E.; Venturelli, D.; Frank, J. Comparing and Integrating Constraint Programming and Temporal Planning for Quantum Circuit Compilation. In Proceedings of the International Conference on Automated Planning and Scheduling, Delft, The Netherlands, 24–29 June 2018; pp. 366–374.
18. 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.
19. Zulehner, A.; Paler, A.; Wille, R. An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *38*, 1226–1236. [CrossRef]
20. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*; Cambridge University Press: Cambridge, UK, 2010.