Quantum Information and Computation, Vol. 18, No. 13&14 (2018) 1095–1114

A METHOD FOR SYNTHESIS AND OPTIMIZATION FOR LINEAR NEAREST NEIGHBOR QUANTUM CIRCUITS BY PARALLEL PROCESSING

ZONGYUAN ZHANG
School of Computer Science and Technology, Nantong University
Nantong, Jiangsu 226001, China

ZHIJIN GUAN[a]
School of Computer Science and Technology, Nantong University
Nantong, Jiangsu 226001, China

HONG ZHANG
Department of Radiation Oncology, School of Medicine, Indiana University
Indianapolis, IN 46202-3082, USA

HAIYING MA
School of Computer Science and Technology, Nantong University
Nantong, Jiangsu 226001, China

WEIPING DING
School of Computer Science and Technology, Nantong University
Nantong, Jiangsu 226001, China

Received March 7, 2018
Revised September 21, 2018

In order to realize the linear nearest neighbor(LNN) of the quantum circuits and reduce the quantum cost of linear reversible quantum circuits, a method for synthesizing and optimizing linear reversible quantum circuits based on matrix multiplication of the structure of the quantum circuit is proposed. This method shows the matrix representation of linear quantum circuits by multiplying matrices of different parts of the whole circuit. The LNN realization by adding the SWAP gates is proposed and the equivalence of two ways of adding the SWAP gates is proved. The elimination rules of the SWAP gates between two overlapped adjacent quantum gates in different cases are proposed, which reduce the quantum cost of quantum circuits after realizing the LNN architecture. We propose an algorithm based on parallel processing in order to effectively reduce the time consumption for large-scale quantum circuits. Experiments show that the quantum cost can be improved by 34.31% on average and the speed-up ratio of the GPU-based algorithm can reach 4 times compared with the CPU-based algorithm. The average time optimization ratio of the benchmark large-scale circuits in RevLib processed by the parallel algorithm is 95.57% comparing with the serial algorithm.

Keywords: linear nearest neighbor(LNN), matrix representation, parallel processing

Communicated by: R Cleve & J Eisert

[a]Email: guan.zj@ntu.edu.cn

1 Introduction

Quantum computing is a new computing technique which achieves high efficiency for some complex computations. With the development of quantum computing, further physical limitations need to be considered. The limited interaction distance between the gate qubits is one of the most common limitations. In particular, quantum computing technics such as ion traps [1, 2], quantum dots [3], and superconductors [4, 5] only allow the quantum gates to perform on adjacent qubits, which is known as the linear nearest neighbor(LNN) constraint.

Since the quantum circuit algorithms are designed without considering this limitation, they should be converted into the LNN architecture. This process can be realized by the insertion of the SWAP gates. However, the number of inserted SWAP gates has a significant impact on the overall quantum costs of the resulting circuit. Accordingly, many approaches attempt to use a minimum number of SWAP gates to realize the LNN architecture. In [6, 7], exact approaches that generate an optimal number of SWAP gates are discussed. Nevertheless, there is a trade-off between the running time and the quantum cost. In [1, 8, 9, 10], heuristic approaches that do not guarantee an optimal solution are proposed. However, to the best of our knowledge, the running time of these solutions still has room for improvement, especially for the conversion of larger circuits. A possible way to reduce time complexity is to accelerate the algorithm by parallel computing.

However, not all algorithms are fit for a parallel computing architecture. Only data-intensive computations [11] without strong data associations are good candidates. Quantum gates in a quantum circuit have strong interactions with each other. This paper intends to break this limitation and solve the LNN problem through parallel processing.

This paper presents a parallel computing algorithm to accelerate the realization of the LNN architecture using the parameter random access memory(PRAM) [12] model. In Section 2, the basic knowledge of the quantum circuits and the PRAM model is expounded. Section 3 proposes an algorithm to achieve the LNN architecture by adding the SWAP gates. To adapt to parallel computing, this method uses a matrix to represent the structure of the quantum circuit. By multiplying matrices of different parts of the whole circuit, a single non-neighbored CNOT gate reaches the LNN and the redundancy SWAP gates are eliminated. Thus, the quantum circuit is converted to the LNN architecture and the quantum cost of the circuit is reduced. Section 4 gives the LNN algorithm based on the PRAM model. The algorithm divides up a circuit evenly into different groups and each group contains two adjacent quantum gates. The SWAP gate adding algorithm is applied in each group. In Section 5, the experiments using the proposed approach are presented. The experimental results show that the proposed algorithm reduces the nearest neighbor cost and quantum cost of the quantum circuit without changing the original function of the circuit. Furthermore, the algorithm has a significant effect on the reduction of runtime for larger scale circuits.

2 Background

2.1 Quantum gates and quantum circuits

The quantum gate is one of the basic elements of quantum circuits. Quantum gate is reversible. The number of the input qubits and the output qubits are equal and there is a one-to-one mapping relationship between the input value and the output value.

An arbitrary reversible 2-qubit quantum logic gate is shown in Figure 1. The control bit of the reversible logic gate is on the i-th line of the quantum circuit and the target bit is on the j-th line of the quantum logic circuit [19].

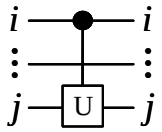

Fig. 1. An arbitrary reversible 2-qubit quantum gate.Here • denotes a control bit and U denotes a specific unitary quantum operation [2].

The CNOT gate [20] is a type of reversible quantum logic gate with two inputs and two outputs. If the input value of the control bit is 0, the output value of the target bit does not change. If the input value of the control bit is 1, the output value of the target bit is the XOR of the two input qubits. The diagram of a CNOT gate is shown in Figure 2.

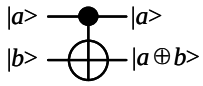

Fig. 2. A CNOT gate. Here • denotes a control bit and $\oplus$ denotes an XOR operation. a and b denote two input values.

The SWAP gate [20] has two target bits and no control bit. The function of SWAP gate is to interchange two input values. The diagram of a SWAP gate is shown in Figure 3.

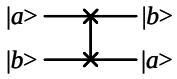

Fig. 3. A SWAP gate. Here $|a>$ and $|b>$ denote the values of input and output of a quantum circuit. $\times$ denotes a swap operation.

Quantum logic circuit is a model for quantum computation [20]. Quantum gates are interconnected by quantum wires in a quantum logic circuit. As the number of inputs and outputs of each quantum gate is the same, any cut through the quantum circuit crosses the same number of wires. Quantum computation is performed by a sequence of quantum gates which are reversible transformations on a quantum mechanical analog of an n-bit register in a quantum logic circuit.

The relationship between the input and output of a quantum logic circuit can be expressed by a linear function [13, 14]. The quantum gate is in LNN state if the control bit and target bit are on adjacent lines in a quantum logic circuit [6].

In an LNN circuit, only adjacent qubits can interact with each other. The distance between the control bit and the target bit in a two-qubit reversible quantum gate is called the nearest neighbor cost (NNC)[9]. The quantum gate with the control bit on the i-th line and the target bit on the j-th line shown in Figure 1 has an NNC of $j - i - 1$. The NNC of the quantum logic circuit is the sum of NNCs of all reversible gates.

Quantum cost [7] of a quantum logic circuit refers to the total quantum cost of each quantum gate in the circuit. The quantum cost of a quantum gate is determined by the type of a gate. In general, the quantum cost of a CNOT gate is 1 and the quantum cost of a SWAP gate is 3 [10].

2.2 The PRAM model

The Parallel Random Access Machine (PRAM)[15] model is a model with shared memory in a Multiple Instruction Stream Multiple Data Stream (MIMD) parallel machine. As shown in Figure 4, it assumes that a device is equipped with unlimited amounts of shared memory and multiple processors with the same function. Any processor has access to the shared memory at any time.

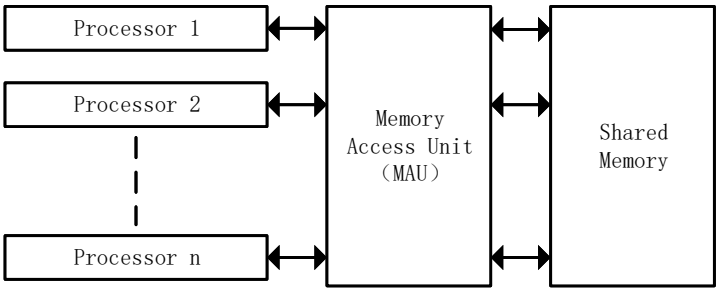

Fig. 4. The structure of a PRAM model.

To solve the LNN problem of quantum logic circuits, we can use the PRAM model to divide a specific quantum circuit into several modules evenly for parallel processing.

3 The LNN realization based on matrix representation

3.1 Relationship between quantum gates

The function of a SWAP gate is equivalent to the function of three CNOT gates arranged in a certain order. Two equivalent quantum logic circuits are shown in Fig.5.(a) and Fig.5.(c). Two adjacent SWAP gates on the same lines can be eliminated as shown in Fig.5.(b).

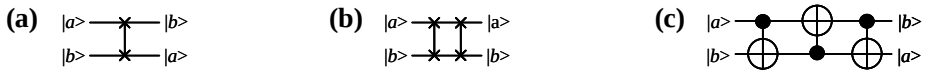

Fig. 5. (a) denotes a SWAP gate. (b) denotes two adjacent SWAP gates in a quantum circuit.(c) denotes three LNN CNOT gates in a quantum circuit. $|a>$ and $|b>$ denote the values of input and output of a quantum logic circuit.

In Fig.5.(a), a and b are the value of input qubits. The vector of input values is set as $|a, b>$. After the operation of three CNOT gates, the order of output value is $|b, a>$ which is equivalent to the function of the SWAP gate. The equivalence is shown in Eq. (1).

$$\begin{aligned} & |a, b> \\ \rightarrow & |a, b \oplus a> \\ \rightarrow & |a \oplus (b \oplus a), b \oplus a> = |b, b \oplus a> \\ \rightarrow & |b, (b \oplus a) \oplus b> = |b, a> \end{aligned} \tag{1}$$

Therefore, the quantum logic circuits shown in Fig.5.(a) and Fig.5.(c) are equivalent in function.

Definition 1: If two quantum gates are adjacent in a quantum circuit, we define the location of two qubits which are on lower lines as L_1 and L_2 respectively and define the location of two qubits which are on higher lines as H_1 and H_2 respectively. As shown in Figure 6, $L_1 = i$, $L_2 = i + 1$, $H_1 = j$, $H_2 = j + 1$.

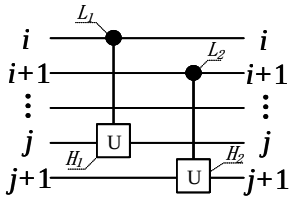

Fig. 6. Two adjacent quantum gates in a quantum circuit.

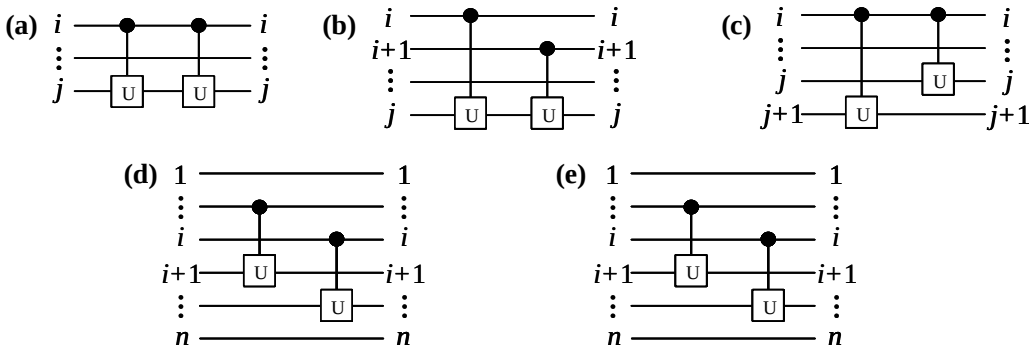

Fig. 7. Different cases of overlapping gates. (a) denotes case 1. (b) and (c) denote case 2.(i)and case 2.(ii). (d) and (e) denote case 3.

If the NNC of two adjacent quantum gates are non-zero and the two adjacent quantum gates satisfy one of the following cases shown in Figure 7, they are overlapping.

1. $L_1 = L_2$ & $H_1 = H_2$
2. (i) $|L_1 - L_2| = 1$ & $H_1 = H_2$
   (ii) $|H_1 - H_2| = 1$ & $L_1 = L_2$
3. $|L_2 - H_1| = 1$ & $|L_1 - H_2| = 1$

3.2 The matrix representation of the quantum circuit structure

The linear structure of some special quantum circuits can be represented by a square matrix. When a set of qubits passes through an empty quantum circuit, the quantum circuit does not have any effect on the qubits. Since the input data is equal to the output data, the empty quantum circuit can be represented by an identity matrix. The order of the matrix is equal to the number of lines of the quantum circuit.

Similar to this representation, the matrix representation of a section of the quantum circuit containing a single quantum gate (a CNOT gate or a SWAP gate) is universal.

Definition 2: The matrix representation of a section of the quantum circuit containing a single CNOT gate is defined as the matrix representation of the quantum circuit for the CNOT gate (MQC).

Definition 3: The matrix representation of a section of the quantum circuit containing a single SWAP gate is defined as the matrix representation of the quantum circuit for the SWAP gate (MQS).

Let n denote the total number of lines in a quantum circuit. Let $i(i \geq n)$ denote the line number of the control bit and $j(j \geq n)$ denote the line number of the target bit of the CNOT gate. Set the element in the i-th column and the j-th row of an n-order identity matrix to 1 and the resulting square matrix is the MQC.

Conversely, if the MQC is given, two qubit positions of the CNOT gate can be located by the element which is non-zero and not on the diagonal of the matrix.

Two operation lines of the matrix correspond to the control bit and the target bit of the CNOT gate separately. In order to distinguish two CNOT gates which are in different orders, we classify two types of CNOT gates by order.

Fig. 8. (a) denotes a CNOT-down gate.(b) denotes a CNOT-up gate.Both (a) and (b) are CNOT gates located in n-line circuits.

Definition 4: Number the matrix from top to bottom by row. Two operation lines are i and j separately. If the target bit of the CNOT gate is below the control bit, the CNOT gate is defined as the CNOT-down gate. Otherwise, the CNOT gate is defined as the CNOT-up gate. The diagrams of the CNOT-up gate and the CNOT-down gate are shown in Figure 8.

Theorem 1: If two non-zero elements are on the same column of the matrix and on the adjacent rows of the matrix, the CNOT gate is in the LNN state. Otherwise, it is not in the

LNN state. Figure 9 shows the diagrams of the LNN CNOT gates.

Fig. 9. (a) denotes a LNN CNOT-down gate. (b) denotes a LNN CNOT-up gate. Both (a) and (b) are CNOT gates located in n-line circuits.

Proof: Since the CNOT-down gate and the CNOT-up gate cover all kinds of CNOT gate, we prove the MQCs are in LNN state.

Number the lines of the quantum circuit from the top down. According to the MQC, the control bit of the CNOT-down gate shown in Fig.9.(a) is on the $(j-1)$-th row and target bit is on the j-th row. The matrix representation is converted from the n-order identity matrix where the element on the $(j-1)$-th column and j-th row is set to 1. Thus, the LNN MQC is shown in Eq. (2).

$$A_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & \underline{1} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2}$$

We can conclude from matrix A_n that two non-zero elements are in the same column of the matrix and these two non-zero elements are located in adjacent rows.

Similarly, the LNN MQC is shown in Eq. (3).

$$B_n = \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & \underline{1} & 0 \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

From matrix B_n, we can conclude that two non-zero elements are in the same column of the matrix and they are in adjacent rows.

In summary, two elements in the same column of the matrix are in adjacent rows, and vice versa.

Theorem 1 is proved □.

Theorem 2: A 2-order MQS can be expressed as a matrix, $S = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$.

Proof: The MQS can be calculated by multiplying three MQCs according to Eq. (1) given in Section 3.1. The linear operation of the quantum circuit in Fig.5.(c) is shown in Eq. (4). Three MQCs are G_1, G_2 and G_3 in sequence. The matrix multiplication represents the process of the

linear input of qubits passing through a sequence of adjacent quantum gates in a quantum circuit. The order of the matrix multiplication is opposite to the order of quantum gates located on the circuit.

$$G = G_3 * G_2 * G_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{4}$$

We can conclude from Eq. (4) that the MQS can be expressed as a symmetric matrix $S = G = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ in a two-qubit LNN quantum circuit.

Theorem 2 is proved □.

Theorem 3: Let n denote the total number of lines in a quantum circuit. If two input qubits of a SWAP gate are on the $i-th(i \geq n)$ and the $(i+1)-th((i+1) \geq n)$ line of the quantum circuit respectively, the n-order MQS can be realized by modifications to an n-order identity matrix as shown in Eq. (5). The 2-order submatrix started from the i-th column and i-th row is equal to the 2-order MQS given in Theorem 2.

$$S_n = \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & \underline{0} & \underline{1} & 0 \\ 0 & \cdots & \underline{1} & \underline{0} & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Proof: Since the two qubits of the SWAP gate are in the LNN state, the equivalent circuit contains three LNN CNOT gates. According to Eq. (1), two quantum circuits shown in Fig.10.(a) and Fig.10.(b) are equivalent.

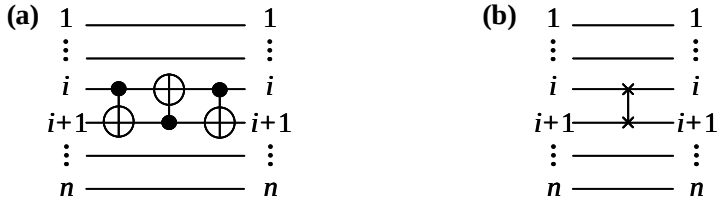

Fig. 10. (a) denotes three LNN CNOT gates located in a n-line quantum circuit. (b) denotes a SWAP gate located in a n-line quantum circuit.

Thus, we multiply three LNN MQCs, G_1, G_2 and G_3, to calculate the matrix representation of the SWAP gate shown in Fig.10.(b).

$$G = G_3 * G_2 * G_1 = \begin{bmatrix} 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 1 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

It can be seen that the matrix representation obtained by Eq. (6) is equal to the MQS proposed by the matrix construction rule in Theorem 3.

Therefore, any n-order ($n \geq 3$) MQS can be expressed as a matrix containing a 2-order symmetric matrix $S = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ and the rest of the diagonal elements in the matrix are all 1.

Theorem 3 is proved □.

3.3 The LNN realization algorithm based on matrix representation

In order to realize LNN without changing the function of quantum circuits, we can scan each quantum gate in the quantum circuit and calculate the NNC of each quantum gate. If the NNC of a CNOT gate is 0, skip it. Otherwise, we convert the quantum gate to LNN state by adding SWAP gates.

Theorem 4: For a 3-line quantum circuit, adding SWAP gates from the top down and from the bottom up are two ways of realizing LNN. As long as the numbers of added SWAP gates are the same, two ways in different order of adding SWAP gates as shown in Fig.11.(b) and Fig.11.(c) are equivalent.

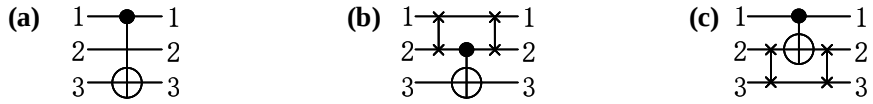

Fig. 11. (a) denotes a CNOT gate. (b) denotes a LNN CNOT gate realized by adding swap gates from the top down. (c) denotes a LNN CNOT gate realized by adding swap gates from the bottom up.

Proof: According to Theorem 2 and Theorem 3, we can prove the equivalence of three quantum circuits shown in Figure 11.

The MQC in Fig.11.(a). is expressed in Eq. (7)

$$G_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{7}$$

In Eq. (7), two non-zero elements are located in the first column. To convert two qubits of the CNOT gate shown in Fig.11.(a) to the LNN state, we can move the qubit from the first line to the second line and add the SWAP gates to both sides of the CNOT gate as shown in Fig.11.(b) and Fig.11.(c).

The linear operation in Fig.11.(b) is shown in Eq. (8).

$$G_b = G_3 * G_2 * G_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{8}$$

Similarly, the linear operation in Fig.11.(c) is shown in Eq. (9).

$$G_c = G_3 * G_2 * G_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{9}$$

Since $G_a = G_b = G_c$, the two ways of adding the SWAP gates do not change the original function of the quantum circuit.

Theorem 4 is proved □.

Since the method proposed in Theorem 4 is universal, the two ways of adding the SWAP gates are applicable to any arbitrary n-line($n \geq 3$) quantum circuit as shown in Figure 12.

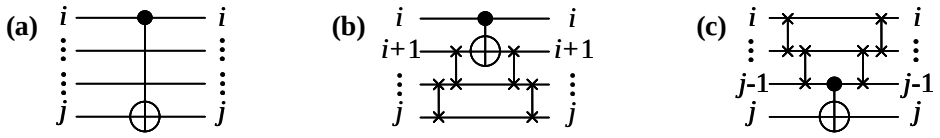

Fig. 12. (a) denotes a CNOT gate. (b) denotes a LNN CNOT gate realized by adding the swap gates from the top down. (c) denotes a LNN CNOT gate realized by adding the swap gates from the bottom up.

To convert a quantum circuit to the LNN architecture, we can add the SWAP gates into the circuit. The LNN architecture is realized by the matrix multiplication of the CNOT gates and the SWAP gates. To minimize the SWAP gates used for the architecture, this section proposes an algorithm to eliminate redundancy SWAP gates.

Theorem 5: If two adjacent quantum gates are overlapping, they have redundancy SWAP gates to be eliminated between them after applying Theorem 4 to both quantum gates. The exact number of the SWAP gates that can be eliminated are given below by $Gate\_Num$ using the representing method of Definition 1.

1. Case 1 of Definition 1 :
   $Gate\_Num = (H_1 - L_1 - 1) * 2$
2. Case 2 of Definition 1 :
   (i) $if \quad |H_1 - L_1| < |H_2 - L_2| \quad then$
   $Gate\_Num = (H_1 - L_1 - 1) * 2$
   (ii) $if \quad |H_1 - L_1| > |H_2 - L_2| \quad then$
   $Gate\_Num = (H_2 - L_2 - 1) * 2$
3. Case 3 of Definition 1 :
   $Gate\_Num = 2$

Proof: If two SWAP gates are inserted into the same line of a quantum circuit successively, the functions of the two SWAP gates can cell out as shown in Fig 5.(b). To maximize the number of the SWAP gates that can be eliminated, the insertion of the SWAP gates should be in different orders for different cases.

In Theorem 5.1, both quantum gates are decomposed by inserting the SWAP gates from the top down as shown in Figure 13. Thus, all the SWAP gates between two original quantum gates can be eliminated layer by layer. The number of the SWAP gates that can be eliminated is $(H_1 - L_1 - 1) * 2$.

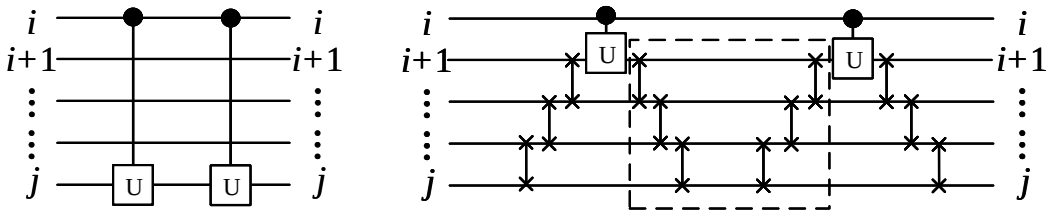

Fig. 13. Two adjacent quantum gates decomposed by inserting SWAP gates.These quantum gates satisfy case 1 of Definition 1.

In Theorem 5.2, since the qubits which are on the lower lines and the higher lines may not be on the same line, we have to scan each line of the circuit to decide the order of inserting the SWAP gates. If $H_1 = H_2$, we insert the SWAP gates from the bottom up. If $L_1 = L_2$, we insert the SWAP gates from the top down. The maximum number of the SWAP gates that can be eliminated is determined by the shorter distance between the two qubits of a quantum gate. Figure 14 shows the process of decomposition of the CNOT gates which satisfy Theorem 5.2 (i). In this case, the number of the SWAP gates that can be eliminated is $(H_1 - L_1 - 1) * 2$. Similarly, the number of the SWAP gates that can be eliminated in Theorem 5.2 (ii) is $(H_2 - L_2 - 1) * 2$.

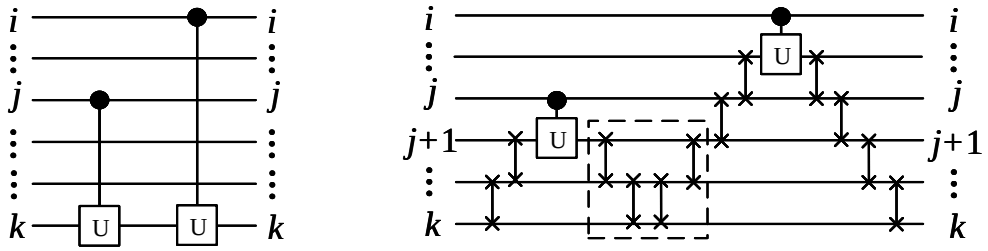

Fig. 14. Two adjacent quantum gates decomposed by inserting the SWAP gates. These quantum gates satisfy case 2 of Definition 1. Since $H_1 = H_2$, the SWAP gates are added from the bottom up.

In Theorem 5.3, the overlapping distance of two adjacent quantum gates only covers two lines of the circuit. The number of the SWAP gates that can be eliminated is 2. Figure 15 shows the process of decomposition of the CNOT gates.

Theorem 5 is proved □.

The algorithm of the LNN realization of two adjacent quantum gates in a quantum circuit is given in Theorem 5. We use Definition 1 to describe the data structure of the algorithm. H_1, H_2, L_1, L_2 represent the line number of the qubits of two quantum gates. We assure $L_1 < H_1$ and $L_2 < H_2$ when entering information of the quantum circuit.

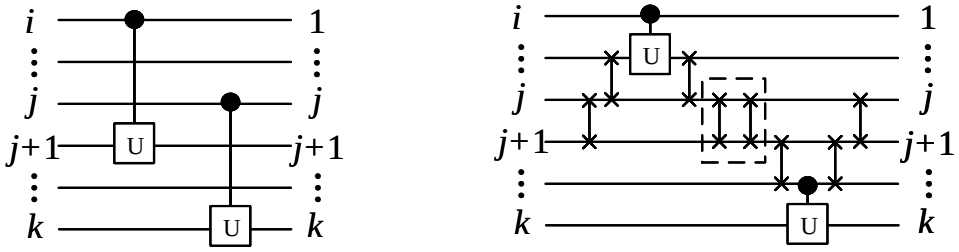

Fig. 15. Two adjacent quantum gates decomposed by inserting the SWAP gates. These quantum gates satisfy case 3 of Definition 1.

Algorithm 1 The SWAP gate adding algorithm: *Gate_Num_Add*( )

Input: Variables:L_1, H_1, L_2, H_2;
Output: *Gate_Num* ;

$t \leftarrow (H_1 - L_1 + H_2 - L_2 + 2) * 2$;
if $H_1 - L_1 = 1$ *or* $H_2 - L_2 = 1$ **then**
  $c \leftarrow 0$;
else if $L_2 - H_1 = 1$ *or* $L_1 - H_2 = 1$ **then**
  $c \leftarrow 2$;
else if $H_1 = H_2$ *and* $|L_1 - L_2| = 1$ **then**
  if $(H_1 - L_1) < (H_2 - L_2)$ **then**
    $c \leftarrow (H_1 - L_1 - 1) * 2$;
  else
    $c \leftarrow (H_2 - L_2 - 1) * 2$;
  end if
else if $L_2 - H_1 = 1$ *or* $L_1 - H_2 = 1$ **then**
  if $(H_1 - L_1) < (H_2 - L_2)$ **then**
    $c \leftarrow (H_1 - L_1 - 1) * 2$;
  else
    $c \leftarrow (H_2 - L_2 - 1) * 2$;
  end if
else if $(H_1 = H_2$ *and* $L_1 = L_2)$ *or* $(L_1 = H_2$ *and* $H_1 = L_2)$ **then**
  $c \leftarrow (H_1 - L_1 - 1) * 2$;
end if
$Gate\_Num \leftarrow t - c$;

The proposed algorithm traverses the circuit to locate the lower number and higher number of the lines where the qubits locate on. If the two qubits are both located, the algorithm is terminated. Thus, the algorithm is a linear search algorithm and the time complexity is determined by the number of circuit lines. In the worst case, the lower qubit is on the first line of the circuit and the higher qubit is on the last line. The algorithm will traverse each line of the circuit and the time consumption is $\mathcal{O}(M)$ (M is the number of lines in a quantum circuit).

4 The LNN realization algorithm based on PRAM model

The matrix-based LNN algorithm proposed in Section 3 can transform two adjacent quantum gates into the LNN state. Thus, a quantum circuit can be divided up evenly into different groups and each group contains two adjacent quantum gates. N is the total number of quantum gates in a quantum circuit. If N is even, the total number of groups is $N/2$. If N is odd, the total number of groups is $(N-1)/2$ and the last quantum gate only performs the SWAP gate adding operation. The LNN realization of quantum gates in each group is considered as a sub-problem and the algorithm proposed in Section 3 is applied to each sub-problem. All sub-problems are processed in parallel.

By adding the SWAP gates, two adjacent quantum gates are converted to the LNN architecture. The algorithm only considers two adjacent quantum gates and has no effect on other quantum gates in the circuit. We use Definition 1 to distinguish each group of quantum gates and calculate the number of the SWAP gates that can be eliminated. We use Theorem 5 to get the maximum number of the SWAP gates that can be inserted. The final result of each group can be obtained by subtracting the number of redundancy SWAP gates.

Algorithm 2 The LNN realization algorithm based on PRAM model

Input: The set of qubit locations of the quantum gates, P_n;
Output: The sum of added swap gates, $Gate\_Num\_Sum$;
1: Divide the given circuit evenly and allocate a structure array C_n to save the data with the help of P_n;
2: Allocate a structure array D_n on a parallel computing device;
3: $D_n \leftarrow C_n$;
4: Run $Gate\_Num\_Add(\quad)$ with the help of P_n on each parallel computing unit;
5: Return $Gate\_Num\_Sum$ to host;

Example 1: The quantum circuit is shown in Figure 16. We apply the proposed method based on the PRAM model to the circuit.

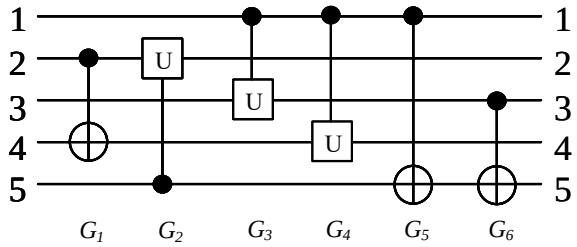

Fig. 16. A 5-line quantum circuit with 6 basic two-qubit quantum gates.

Step 1: Divide the initial circuit into groups and each group contains two quantum gates. In the circuit shown in Figure 16, we divide the entire circuit into three groups and put each group into a parallel processing unit, as shown in Figure 17.

Step 2: Perform the matrix-based LNN algorithm on each group. For example, a matrix-based LNN algorithm is applied to G_1 and G_2 in the first group respectively. Since G_1 and G_2

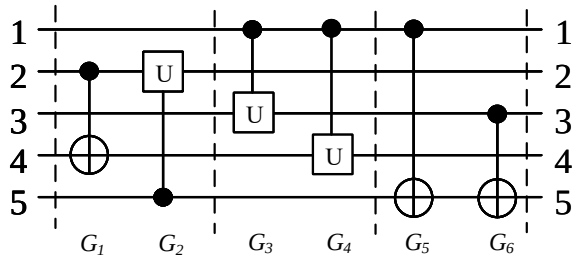

Fig. 17. A 5-line quantum circuit with 6 basic two-qubit quantum gates. The dotted lines divide the circuit into 3 different groups. Each group contains two quantum gates.

satisfy case 2 of Definition 1, we add the SWAP gates from top to bottom. The steps of adding and reducing SWAP gates in different groups are shown in Fig.18.(a), (b), (c) separately.

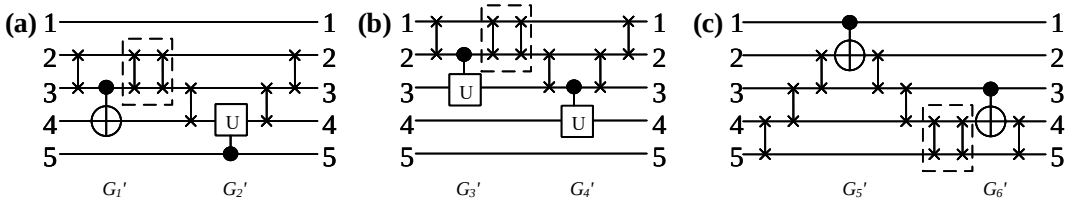

Fig. 18. The LNN realization process of each group of quantum gates. The elimination rules in Theorem 5 are applied. G_n' is the LNN state of G_n in the original circuit.

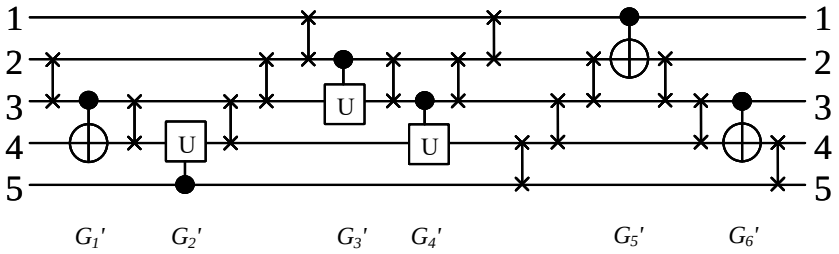

Fig. 19. The LNN quantum circuit converted from the original circuit by adding the SWAP gates. G_n' is the LNN state of G_n in the original circuit.

The LNN quantum circuit converted from the original circuit in Figure 16 is shown in Figure 19.

The equivalence of the final quantum circuit and the original one is proved below.

Divide the quantum circuit into six groups along the dotted lines as shown in Figure 20.

As shown in Figure 21, by the change of line number, the final output sequence is equivalent to the input sequence. The optimized quantum circuit is equivalent to the original circuit in function.

N is the number of quantum gates in the circuit. M is the number of lines. The algorithm needs to execute $N/2$ cycles and the running time of each cycle is $\mathcal{O}(M)$. If the algorithm

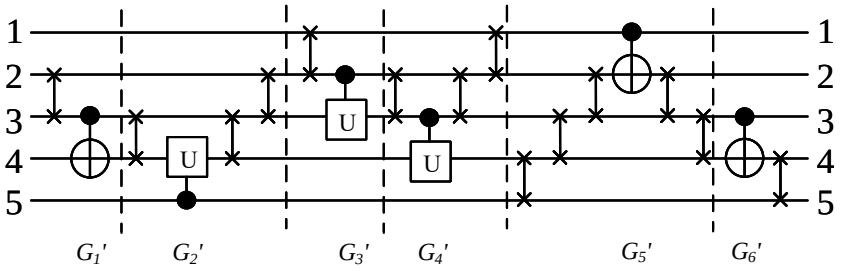

Fig. 20. The LNN quantum circuit converted from the original circuit by adding the SWAP gates. The dotted lines divide the circuit into 6 different groups. Each group represents part of the circuit which has the same function with the corresponding quantum gate in the original circuit.

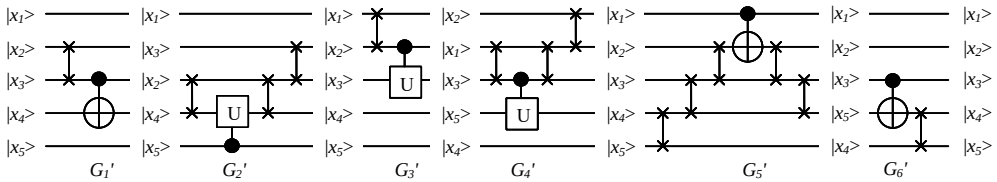

Fig. 21. The sequence of the input values after applying operations to 6 original quantum gates. Here the output sequence is the same as the input sequence.

is executed serially, the time complexity $\mathcal{O}(M * N/2)$. If the algorithm is designed based on the PRAM model, each loop body is put into a parallel processor. Therefore, the time complexity of the algorithm performed on the parallel processing device is $\mathcal{O}(M)$. The time consumption of the transferring data between the parallel processor and the host is $\mathcal{O}(N)$. Therefore, the algorithm based on the PRAM model has a time complexity of $\mathcal{O}(M + N)$. Consequently, the speed-up ratio of the parallel algorithm is $\mathcal{O}(M * N)/\mathcal{O}(M + N)$ comparing to the conventional serial one.

RevLib[16] is a benchmark library of quantum circuits. The largest circuit in RevLib is *cpu_alu_32bit_425.real*. The number of the lines exceeds 6000 and the number of the gates in the circuit exceeds 30000. Therefore, M is 6000 and N is 30000 in extreme cases, and the speed-up ratio is $\mathcal{O}(6000 * 30000)/\mathcal{O}(6000 + 30000)$, that is, the maximum speed-up ratio is 5000.

5 Experimental results

In this section, the experiments using the proposed approach are presented. The programs are carried out on a GeForce GT 630M graphics card with 1 GB RAM, installed in a PC with an Intel Core i5-3317U CPU with 1.70GHz and 8 GB of memory running the Windows 10. We design the experiments from two aspects, the reduction ratio of quantum cost and the running time of the program. To evaluate the performance of the proposed solution, quantum circuits from RevLib have been applied. All quantum circuits are decomposed into two-qubit quantum circuits using the method proposed in [17, 18].

The lines of quantum circuits ranged from $3 \sim 8$ are included to evaluate the optimizations of the quantum cost. The number of the decomposed quantum gates in each quantum circuit is

less than 400. Experiments show that our algorithm has a distinct advantage on the reduction of quantum cost, especially for groups applied to case1 and case2 in Theorem 5. It can be seen that using our approach fewer SWAP gates are implemented and therefore the reduction of quantum cost is improved by 34.3% in average compared to [9], except for a few of the cases in which the elimination algorithm is limited.

In Table 1, *Benchmark* represents the name of quantum circuit. N is the number of lines. *Original_qc* represents the initial quantum cost of a circuit. *Line_Order_qc* shows the optimized quantum using the approach proposed in [9]. The quantum cost of our method and the optimization comparing with [9] are shown in *Cuda_qc* and *Qc_optimization* separately.

Table 1. Results of quantum cost optimizations for quantum circuits between $3 \sim 8$ line.

benchmark	n	Original_qc	Line_Order_qc[9]	Cuda_qc	Qc_optimization
3_17_13	3	14	26	22	15.38%
hwb4_52	4	23	63	47	25.40%
decod24_v3_46	4	9	21	23	0
4_49_17	4	32	92	50	45.65%
4gt5_75	5	21	70	56	20%
4gt11_84	5	7	14	21	0
4gt10_v1_81	5	34	120	100	16.67%
4gt13_v1_93	5	16	74	34	54.05%
hwb5_55	5	104	335	180	46.27%
4mod5_v1_23	5	24	72	40	44.44%
4mod7_v0_95	5	38	121	56	53.72%
aj_e11_165	5	45	160	96	40%
alu_v4_36	5	31	98	90	8.16%
4gt4_v0_80	6	34	132	80	39.39%
4gt12_v1_89	6	42	141	150	0
hwb6_58	6	142	542	200	63.10%
mod5adder_128	6	83	330	200	39.39%
mod8_10_177	6	88	317	210	33.75%
ham7_104	7	83	327	102	68.81%
rd53_135	7	77	303	156	48.51%
hwb7_62	8	2325	12853	5430	57.75%
Ave_qc			34.30%		

The lines of quantum circuits ranged from $8 \sim 6205$ are included to evaluate the optimizations of the running time for large-scale quantum circuits. Based on the time complexity of the two methods given in Section 3 and Section 4, we give the theoretical value of the optimization rate of the parallel algorithm. We analyse the benchmark large-scale circuits in RevLib. For each circuit, we give the theoretical running time of serial computation and parallel computation proposed in this paper. Analysis results show that the average time optimization ratio of the GPU-based algorithm is 95.57% of the CPU-based algorithm for large-scale quantum circuits.

In Table 2, *Benchmark* represents the name of quantum circuit. N is the number of lines. *Original_qc* represents the initial quantum cost of a circuit. We assume that the execution time of each instruction as a unit time(ut). *Serial*(ut) shows the theoretical running time of

serial computation. *Parallel(ut)* shows the theoretical running time of parallel computation. *Optimization_ratio* shows the time optimization ratio of the parallel computation comparing with the serial computation.

Table 2. Results of time optimization ratio for large scale quantum circuits.

benchmark	n	qc	Serial(ut)	Parallel(ut)	Optimization_ratio
hwb8_118	8	14260	114080	14268	87.49%
urf1_149	9	57770	519930	57779	88.89%
sym9_148	10	4368	43680	4378	89.98%
urf3_155	10	132340	1323400	132350	90.00%
plus63mod4096_163	12	25492	305904	25504	91.66%
plus63mod8192_164	13	32578	423514	32591	92.30%
plus127mod8192_162	13	57400	746200	57413	92.31%
0410184_169	14	90	1260	104	91.75%
ham15_108	15	453	6795	468	93.11%
urf6_160	15	53700	805500	53715	93.33%
cnt3-5_180	16	120	1920	136	92.92%
add8_172	25	96	2400	121	94.96%
add16_174	49	192	9408	241	97.44%
add32_183	97	384	37248	481	98.71%
add64_184	193	768	148224	961	99.35%
apex5_290	1025	10349	10607725	11374	99.89%
frg2_297	1219	12468	15198492	13687	99.91%
bubblesort_32_365	1596	42281	67480476	43877	99.93%
bubblesort_32_437	1597	24569	39236693	26166	99.93%
seq_314	1617	19362	31308354	20979	99.93%
cpu_alu_16bit_352	2140	31244	66862160	33384	99.95%
cpu_alu_16bit_424	2141	30208	64675328	32349	99.95%
cpu_alu_32bit_353	6204	112396	697304784	118600	99.98%
cpu_alu_32bit_425	6205	107136	664778880	113341	99.98%
Ave_opt			95.57%		

The line chart in Figure 22 is employed to show the trend of the optimization ratio of the parallel algorithm. The line chart shows that the larger the scale of the quantum circuit is, the better the optimization ratio is.

CUDA platform uses blocks and threads to represent parallel computing. Generally, we start 128 blocks and each block contains 128 threads. The index of each thread is $threadIdx.x + blockIdx.x * blockDim.x$.

Since the time complexity of our algorithm has no connection with the type of quantum gates, we construct a number of quantum circuits containing random two-qubit quantum gates. The numbers of quantum gates in each circuit range from 400 to 286840. In Figure 23, experiments show that the quantum circuits on a smaller scale run slower on the GPU than on the CPU. It can be explained by the extra time consumption of transferring data between host and device. With the increasing scale of quantum circuits, the advantage of parallel computation becomes clear. With the full use of parallel units in the GPU, the speed-up ratio of the GPU-based algorithm can reach to 4 times comparing with the CPU-

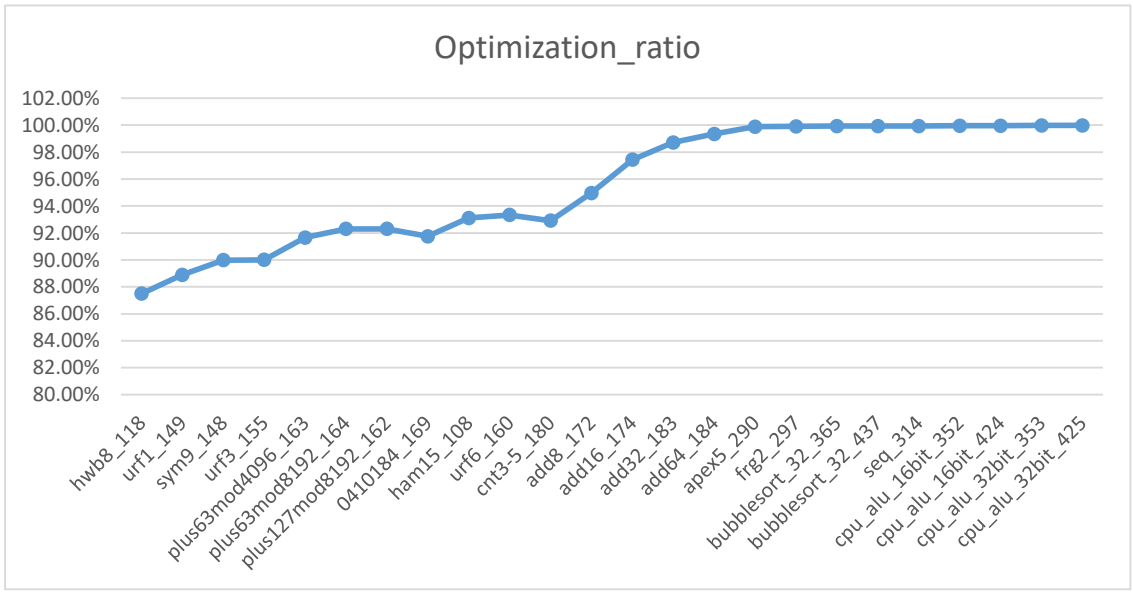

Fig. 22. Time optimization ratio for large scale quantum circuits.

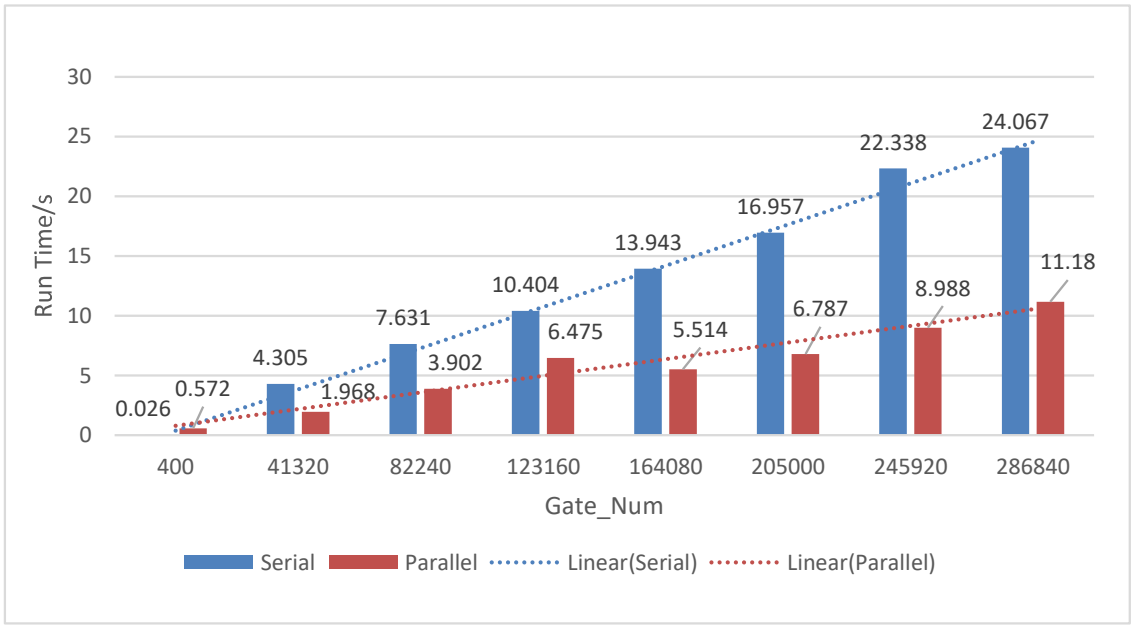

Fig. 23. Results of runtime of serial algorithm and parallel algorithm.

based algorithm. Thus, our algorithm has a significant effect on the reduction of the runtime for larger scale circuits.

6 Discussion and conclusions

In this paper, we propose a linear nearest neighbor optimization algorithm based on matrix transformation of quantum circuits constructed by the PRAM model. Considering the different situations of adjacent quantum gates, we design the algorithm by dividing the circuits into different groups, which can effectively reduce the quantum cost and improve the computational efficiency by inserting the SWAP gates on the parallel computing device. Experimental results show that the algorithm can optimize both quantum cost and computing speed. In addition, the algorithm enables the conversion of large-scale quantum circuits into the LNN architectures in a short period of time.

While our method is designed for nearest neighbor constraints, the recently proposed IBM QX architectures require the satisfaction of slightly different constraints [21]. In the IBM QX architectures, a given quantum circuit need to be decomposed into a sequence of elementary gates [22]. The basic gates in the IBM QX architectures include the H (Hadamard) gate, T (phase shift by $\pi/4$) gate and the CNOT gate in the Clifford+ T library. To satisfy the IBM QX constraints, we can add H-gates and SWAP gates to the quantum circuit. The algorithm proposed in this paper is based on the NCV library and the linear nearest neighbor is realized by adding SWAP gates. Both the Clifford+ T and the NCV library are suitable for our proposed parallel optimization algorithm.

In the further research, we will focus on reducing the quantum cost required to satisfy the constraints of the IBM QX architectures and integrating this method into the parallel computing system to improve the efficiency of the algorithm.

Acknowledgements

This work was supported by Chinese National Natural Science Foundation (61402244), Natural Science Foundation of Jiangsu Province (BK20151274), General Project of Natural Science Research of Jiangsu Higher School (14KJB520033), Postgraduate Research & Practice Innovation Program of Jiangsu Province (KYCX17_1916).

References

1. Kole, Abhoy, K. Datta and I. Sengupta (2016), *A Heuristic for Linear Nearest Neighbor Realization of Quantum Circuits by SWAP Gate Insertion Using N-Gate Lookahead*, IEEE Journal on Emerging & Selected Topics in Circuits & Systems 6.1:62-72
2. Wille, R., Keszocze, O., Walter, M. and Rohrs, P. (2016), *Look-ahead schemes for nearest neighbor optimization of 1D and 2D quantum circuits*, Asia and South Pacific Design Automation Conference (Vol.2074, pp.292-297). IEEE.
3. Jones, N. C., Van Meter, R., Fowler, A. G., McMahon, P. L., Kim, J., Ladd, T. D. and Yamamoto, Y. (2012), *Layered architecture for quantum computing*, Physical Review X, 2(3), 031007.
4. Strauch, F. W., Johnson, P. R., Dragt, A. J., Lobb, C. J., Anderson, J. R. and Wellstood, F. C. (2003), *Quantum logic gates for coupled superconducting phase qubits*, Physical review letters, 91(16), 167005.
5. Ohliger, M. and Eisert, J. (2012), *Efficient measurement-based quantum computing with continuous-variable systems*, Physical Review A, 85(6), 062318.
6. Wille, R., Lye, A. and Drechsler, R. (2014), *Exact reordering of circuit lines for nearest neighbor quantum architecture*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33(12), 1818-1831.

7. Wille, R., Lye, A. and Drechsler, R. (2014), *Optimal SWAP gate insertion for nearest neighbor quantum circuits*, In Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific (pp. 489-494). IEEE.
8. AlFailakawi, M., AlTerkawi, L., Ahmad, I. and Hamdan, S. (2013), *Line ordering of reversible circuits for linear nearest neighbor realization*, Quantum information processing, 12(10), 3319-3339.
9. Saeedi, M., Wille, R. and Drechsler, R. (2011), *Synthesis of quantum circuits for linear nearest neighbor architectures*, Quantum Information Processing, 10(3), 355-377.
10. Shafaei, A., Saeedi, M. and Pedram, M. (2013), *Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures*, In Proceedings of the 50th Annual Design Automation Conference (p. 41). ACM.
11. Middleton, A. M. (2010), *Data-intensive technologies for cloud computing*, In Handbook of cloud computing (pp. 83-136). Springer, Boston, MA.
12. Keler, C. W. and Trff, J. L. (1999), *Language and library support for practical PRAM programming*, Parallel Computing, 25(2), 105-135.
13. MARKOV, K. N. P. I. L. and HAYES, J. P. (2008), *Optimal synthesis of linear reversible circuits*,Quantum Information and Computation, 8(3&4), 0282-0294.
14. Schaeffer, B. (2013), *Computer Aided Design of Permutation, Linear, and Affine-Linear Reversible Circuits in the General and Linear Nearest-Neighbor Models*, Doctoral dissertation, Portland State University
15. Fortune, S. and Wyllie, J. (1978), *Parallelism in random access machines*, In Proceedings of the tenth annual ACM symposium on Theory of computing (pp. 114-118). ACM.
16. Wille, R., Groe, D., Teuber, L., Dueck, G. W. and Drechsler, R. (2008), *RevLib: An Online Resource for Reversible Functions and Reversible Circuits*, International Symposium on Multiple Valued Logic (pp.220-225). IEEE Computer Society.
17. Lye, A., Wille, R. and Drechsler, R. (2015), *Determining the minimal number of swap gates for multi-dimensional nearest neighbor quantum circuits*, In Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific (pp. 178-183). IEEE.
18. Rahman, M. M., Dueck, G. W., Chattopadhyay, A. and Wille, R. (2016), *Integrated synthesis of linear nearest neighbor ancilla-free MCT circuits*, In Multiple-Valued Logic (ISMVL), 2016 IEEE 46th International Symposium on (pp. 144-149). IEEE.
19. Hirvensalo, M. (2004), *Quantum Computing (Natural Computing Series)*,SpringerVerlag.
20. Law J. (2001), *Quantum computation and quantum information[M]*,ACM.
21. Zulehner A, Paler A, Wille R. *An Efficient Methodology for Mapping Quantum Circuits to the IBM QX Architectures[J]*,IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018, PP(99):1-1..
22. Miller D M, Wille R, Sasanian Z. (2001), *Elementary Quantum Gate Realizations for MultipleControl Toffoli Gates[C]*,IEEE International Symposium on Multiple-Valued Logic. IEEE Computer Society, 2011:288-293.