





A Quantum Algorithm for Deciding Graph 2-Coloring Problem in Embedded Systems

Jinfeng He^{1*}, Yao Zhang² and Shanxian Lin³

^{1,2,3} School of Information Science and Technology, Nantong University, Nantong 226000, China
hjf89@ntu.edu.cn, 2565649106@qq.com, 1369362184@qq.com

Correspondence author: Jinfeng He, [hj89@ntu.edu.cn](mailto:hjf89@ntu.edu.cn)

Abstract. The K -coloring of a graph is to assign K colors to each vertex so that the adjacent vertices have different colors. When $K \geq 3$, the K -coloring problem is NP-complete. 2-coloring is considered in the paper. We propose a classical method for any number n , it outputs a quantum circuit that for any graph with n vertices and a coloring strategy, deciding whether the coloring strategy is correct. The method can automatically generate the corresponding quantum circuits according to the scale of the problem, and can be easily extended to K -coloring for $K \geq 3$. The resulting quantum circuit provides a candidate for the oracle of the Grover algorithm that solves the coloring problem.

Key words: graph coloring; quantum circuit; reversible logic circuit

DOI: <https://doi.org/10.14733/cadaps.2024.S8.31-43>

1 INTRODUCTION

1.1 Graph Coloring Problem

The coloring problem of graphs originates from the famous "four-color conjecture" problem, which has been proved in [2], [8]. Four-color conjecture problem is also known as the four-color theorem. It is a famous mathematical theorem. Its content is that "any map with only four colors can dye countries with common boundaries with different colors." That is to say, any map on a plane (or sphere) can only use four colors to dye so that neighboring countries will not have the same color. In mathematical language, the plane is arbitrarily subdivided into non-overlapping areas. Each area can always be marked with one of the four numbers 1, 2, 3, and 4 without making the adjacent two

areas get the same number. The adjacency here means that there is a whole section of boundary between two areas that is common.

On this basis, the coloring problem of a graph abstracts each region in the plane into a point. Connecting two points indicates that the two points are adjacent. If they are not adjacent, they will not be connected. Now, given an undirected graph and K colors, where V is the vertex set and E is the edge set, we need to find a possible coloring strategy to use K colors to color the vertices of the graph, so that any two adjacent vertices do not have the same color. If it can be found, then it is said that the undirected graph can be colored by K colors. In other words, the graph coloring problem is to divide the vertex set V into K groups, each group corresponds to the same color, forming an independent set[14], that is, there are no adjacent vertices. The optimization problem of graph coloring is to obtain the minimum K value. In all these applications, finding the minimum number of colors (K value) required to color the graph optimally is often the goal. This corresponds to minimizing the usage of resources, such as channels, time slots, processors, or registers, which is crucial in the context of resource-constrained Embedded Systems.

The graph coloring problem has a lot of applications. Here, taking the formulation of the exam schedule as an example, the problem can be abstracted into a graph coloring problem, and an undirected graph $G = (V, E)$ is established. The exam subjects are regarded as vertices, and there is an edge between the two exam subjects if and only if one student needs to take the two exams, and the subjects of the same exam at the same time are regarded as a color. Then, the problem of the test schedule can be transformed into the following: make the undirected graph $G = (V, E)$ point coloring number minimum under the condition that the test time does not conflict[17],[22]. In addition, there are frequency allocation, mobile radio frequency distribution, Sudoku, bipartite graphs, Work scheduling[1], etc.

1.2 Classical Algorithms

When $K \geq 3$, the graph coloring problem is an NP-complete problem. The traditional classical algorithms[5],[15],[3] are as follows.

1.Brute force algorithm

This method can generate all possible color combinations(K^V combinations), where K is the number of colors and V is the number of vertices[9]. After color generation, we use the recursive function to check whether any two adjacent vertices have the same color to determine whether the color combination meets the requirements. In this algorithm, we generate a total of K^V color combinations. Therefore, it requires exponential time, and the time complexity is $O(K^V)$.

2.Backtracking algorithm

It is a search algorithm that systematically searches for the solution to the problem[13]. This method can assign colors to each vertex one by one. The coloring will only start from the first index, but before assigning any color, we will first use the recursive function to check whether it meets the constraint that no two adjacent vertices have the same color. If the current color assignment meets the conditions, it will be added to the solution; otherwise, it will be backtracked. Similar to the brute force algorithm, this algorithm also generates a total of K^V color combinations. Therefore, it also requires exponential time, and the time complexity is $O(K^V)$.

3.Greedy algorithm

It always makes the best choice in the current view when solving problems. According to Brooks theorem[11],[27], for any graph G , there is $X(G) \leq d + 1$, where d represents the maximum degree of graph G , $X(G)$ represents the chromatic number of graph G , and the chromatic number is the minimum color number required for graph coloring. If the maximum degree of vertices in a given graph is d , the greedy method to solve the graph coloring problem can use at most $d + 1$ colors. This method cannot guarantee the minimum number of colors but can guarantee the upper limit of the number of colors. We will color the current vertex with the lowest-numbered color. After that, we will check whether the currently assigned minimum color is used for any adjacent vertices of the current vertex. If it is not used, we can continue to process the next vertex, otherwise, we will assign the next color. In this algorithm, we greedily assign a color to each vertex of the graph and check whether the assigned color meets the constraint conditions[24],[19], that is, no two adjacent vertices have the same color.

The calculation time complexity of the above traditional methods is too large to solve the problem in the effective time. Here we open up another method - quantum circuit. This algorithm uses the natural parallel processing ability of the quantum computer to solve the coloring problem of the graph with the quantum algorithm[20],[23],[25],[6],[4],[7]. This algorithm takes the 2-coloring problem of the graph as an application example.

The structure of this paper is as follows. In Section 2, we give the basic definitions and notations of quantum circuits. In Section 3, we give the method that the reversible logic circuits for deciding 2-coloring, and decompose them into equivalent Clifford+T circuits. In Section 4, we summarize and conclude the paper.

2 PRELIMINARIES

The quantum circuit[18] model is the most widely used quantum computing model. It provides a basic framework for the construction of quantum algorithms and the physical implementation of quantum computers. Using the basic ideas of measurement calculus and distributed quantum computing, a measurement quantum circuit model is proposed. It is composed of quantum bits, lines representing the evolution of quantum bits (timelines), and various quantum logic gates acting on quantum bits. In essence, it is an execution sequence of quantum logic gates. It is executed sequentially from left to right. Finally, the quantum measurement is often required to read the results. Unlike traditional circuits, which are connected by metal wires to transmit voltage signals or current signals, in quantum circuits, the circuit is connected by time, that is, the state of quantum bits evolves naturally with time, and is operated according to the instructions of Hamiltonian operators until it meets a logic gate. Since every quantum logic gate that constitutes a quantum circuit is a unitary operator, the whole quantum circuit is also a large unitary operator.

The basic content of quantum computing includes two categories: quantum bits and quantum logic gates. Quantum bits include single quantum, double quantum, and multiple quantum bits. The two basic particles $|0\rangle$ and $|1\rangle$ are used to represent the basic state of quantum bits. In this paper, we use quantum circuits to consider the 2-coloring problem of graphs. So we can use these two states to represent two colors. At the same time, the main logic gates used are as follows.

1. Pauli-X gate

The Pauli-X gate acts on a single quantum bit, which is quantum equivalent to the NOT gate of a classical computer, and is used to flip the quantum state: $|0\rangle$ becomes $|1\rangle$, $|1\rangle$ becomes $|0\rangle$ [21]. In quantum circuits, we use "X" to express (as shown in Figure 1a). Its matrix form is as follows.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

2.CNOT (controlled-NOT) gate

The CNOT gate is a two-qubit gate, which is used to flip and modify the target bit according to the control bit state. Its matrix form is as follows.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In the quantum circuit, we use two lines to represent two qubits (as shown in Fig1b). Among them, the upper line (containing a black circle) is the control qubit, the lower line (containing a cross circle) is the target qubit, and the possible states of the two qubits are $|0\rangle$ or $|1\rangle$. Its truth table is shown in Table 1.

Input		Output	
Control bit	Target bit	Control bit	Target bit
<i>c</i>	<i>t</i>	<i>c</i>	<i>t</i>
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 1: The Truth Table of the CNOT Gate.

It can be seen from the table that the meaning of the CNOT gate is that when the control bit is in the $|0\rangle$ state, the target bit does not change; When the control bit is in the $|1\rangle$ state, the Pauli-X gate (quantum non-gate) operation is performed on the target bit, that is, $|0\rangle$ becomes $|1\rangle$, and $|1\rangle$ becomes $|0\rangle$. It is particularly important to note that the positions of control bits and target bits cannot be exchanged.

3.Toffoli gate

The Toffoli gate is a three-input, three-output reversible logic gate[10],[26]. It has the same usage and principle as the CNOT gate and also changes the target bit according to the control bit status.

The difference is that it acts on three qubits, of which there are two control bits and one target bit. Its matrix form is as follows.

$$Toffoli = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

In the quantum circuit, we use three lines to represent three qubits (as shown in Fig1c). Among them, the upper two lines (containing black circles) are control qubits, and the lowest line (containing cross circles) is target qubits. Its truth table is shown in Table 2.

Input			Output		
Control bit	Target bit		Control bit	Target bit	
<i>c1</i>	<i>c2</i>	<i>t</i>	<i>c1</i>	<i>c2</i>	<i>t</i>
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Table 2: Truth Table of the Toffoli Gate.

It can be seen from the table that the Toffoli gate means that only when the two control bits are in the $|1\rangle$ state, the Pauli-X gate (quantum non-gate) operation is performed on the target bit, that is, $|0\rangle$ becomes $|1\rangle$, $|1\rangle$ becomes $|0\rangle$, and the target bit does not change in other states. It is also important to note that the positions of control bits and target bits cannot be exchanged.

4.MCT gate

The multiple-control Toffoli (MCT) gate has a target line and several control lines[16]. It continues to increase the number of control qubits based on the CNOT gate and the Toffoli gate, with the same principle. Its matrix form is as follows.

$$MCT = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}$$

In quantum circuits, we can use n lines to represent n qubits (as shown in Fig1d). Among them, the top $n-1$ line (containing black circles) are control qubits, and the bottom line (containing a cross circle) is the target qubit. Its truth table is shown in Table 3.

Input					Output				
Control bit				Target bit	Control bit				Target bit
$c1$	$c2$...	cn	t	$c1$	$c2$...	cn	t
0	0	...	0	0	0	0	...	0	0
0	0	...	0	1	0	0	...	0	1
0	0	...	1	0	0	0	...	1	0
0	0	...	1	1	0	0	...	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	1	...	0	1	1	1	...	0	1
1	1	...	1	0	1	1	...	1	1
1	1	...	1	1	1	1	...	1	0

Table 3: The Truth Table of the MCT Gate.

It can be seen from the table that the meaning of the MCT gate is that only when $n-1$ control bits are in the $|1\rangle$ state, the Pauli-X gate (quantum non-gate) operation is performed on the target bit, that is, $|0\rangle$ becomes $|1\rangle$, $|1\rangle$ becomes $|0\rangle$, and the target bit does not change in other states[16].

5.Measurement gate

On a real quantum computer, the information on the final state of the quantum system can only be obtained by measuring the final state. The measurement uses the quantum bit register to measure

it and then outputs the result in the form of classical information. The measurement result is mapped to the classical register that can be effectively represented. The measurement operation is represented by the measurement symbol. It always treats the input as a qubit register (represented as a solid line) and the output as classical information (represented as a double line).

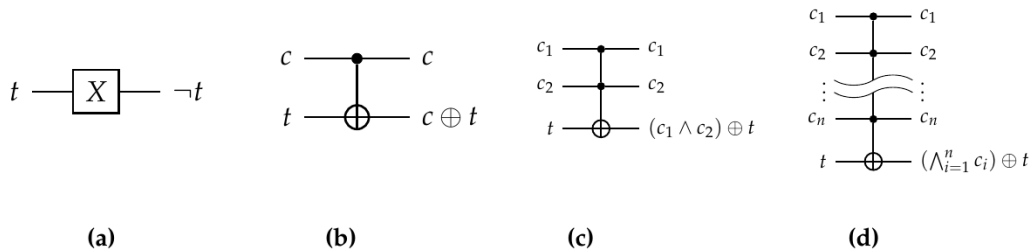


Figure 1: The Illustration of four Kinds of Quantum Gates. (a)The X Gate, (b)The CNOT Gate, (c)The Toffoli Gate, (d)The MCT Gate.

The quantum circuit algorithm uses these logic gates to determine whether the colors of adjacent vertices are equal. Here, only the coloring of two colors are considered, which are represented by $|0\rangle$ and $|1\rangle$ respectively, and satisfy the possible state of quantum circuit bits. In the algorithm, the Qiskit library is used to generate circuits, and the Pillow library (sometimes called the PIL library) is used for image processing. Qiskit is an open-source SDK for quantum computing developed by IBM[12] and has an extremely perfect ecosystem. The Qiskit quantum software development kit accelerates the development of quantum applications by providing a complete set of tools required to interact with quantum systems and simulators, and realizes the simulation analysis of quantum image processing algorithms. The Pillow library is the basic library for Python image processing, which can easily store, display and process images in various formats.

3 2-COLORING PROBLEM

In this section, we propose a classical method for any number n , it outputs a quantum circuit that for any graph with n vertices and a coloring strategy, deciding whether the coloring strategy is correct.

3.1 Quantum Circuit Construction

Algorithm 1 Generate reversible circuits for 2-coloring

Input: The number n of the vertices in the graph.

Output: A reversible circuit to decide the 2-coloring for the graphs with n vertices.

1. function gen(n)
 2. var $v_num, e_num, add_num, x_num$: integer
 3. $v_num \leftarrow n$
 4. $e_num \leftarrow n * (n - 1) / 2$
 5. $add_num \leftarrow 1$
 6. $x_num \leftarrow e_num$
 7. begin
-

```

8. edges = QuantumRegister(e_num, 'e')
9. vertexes = QuantumRegister(v_num, 'v')
10. aids1 = AncillaRegister(e_num, 'a1')
11. aids2 = AncillaRegister(e_num, 'a2')
12. out = QuantumRegister(1, 'o')
13. c = ClassicalRegister(1, 'c')
14. circuit = QuantumCircuit(edges, vertexes, aids1, aids2, out, c)
15. circuit.x(aids2)
16. for i : = 0 to v_num
17. for j : = i + 1 to v_num
18. circuit.ccx(vertexes[i], vertexes[j], aids1[k])
19. circuit.x(vertexes[i])
20. circuit.x(vertexes[j])
21. circuit.ccx(vertexes[i], vertexes[j], aids1[k])
22. circuit.x(vertexes[i])
23. circuit.x(vertexes[j])
24. circuit.ccx(edges[k], aids1[k], aids2[k])
25. circuit.barrier()
26. x_num += 4
27. add_num += 3
28. gate = MCXGate(e_num)
29. ct = list(range(e_num * 2 + v_num, e_num * 3 + v_num + 1))
30. circuit.append(gate, ct)
31. circuit.measure(out, c)
32. end
33. end gen

```

The complexity of the algorithm is $T(n) = n * (n + 1)/2 + 27$, $O(n) = n^2$.

In the algorithm, we count the number of Toffoli gates and Pauli-X gates corresponding to different vertex points. Table 4 shows the number of different gates for several kinds of graphs.

Number of vertices	Number of gates				Depth	Number of qubits
	X	CNOT	Toffoli	MCT		
3	15	0	9	1	30	14
4	30	0	18	1	57	24
5	50	0	30	1	93	37
6	75	0	45	1	138	53
7	105	0	63	1	192	72

Table 4: The Number of Quantum Gates.

The quantum circuit generation is as follows.

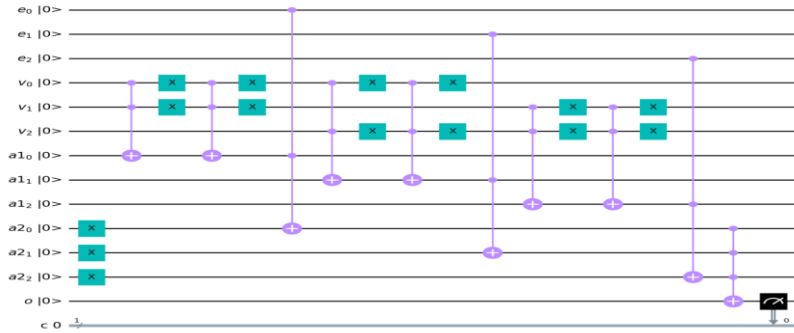


Figure 2: The Quantum Circuit Generated When the Number of Vertices is 3.

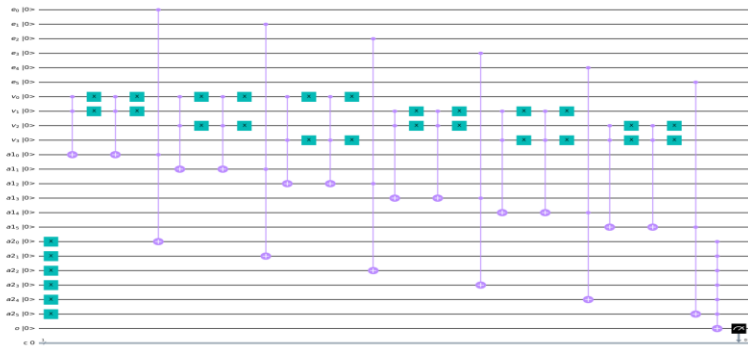


Figure 3: The Quantum Circuit Generated When the Number of Vertices is 4.

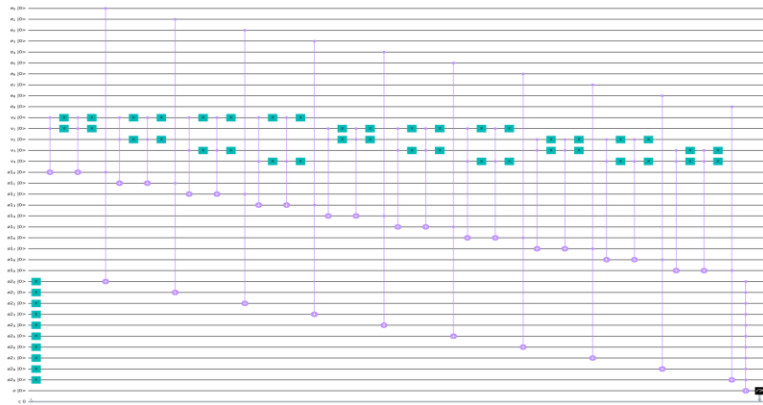


Figure 4:The Quantum Circuit Generated When the Number of Vertices is 5.

Take 2-coloring of three vertices as an example to illustrate the algorithm. Three vertices have a maximum of three edges. e_0 , e_1 , and e_2 are side registers. v_0 , v_1 , and v_2 are vertex registers. a_{10} , a_{11} , a_{12} and a_{20} , a_{21} and a_{22} are auxiliary bit registers. o is the output register. c is a classical bit, and the initial value is $|0\rangle$.

Through the first Toffoli gate, because the control bits v_0 and v_1 are both $|0\rangle$, the target bit a_{10} remains unchanged at $|0\rangle$; v_0 is transformed from $|0\rangle$ to $|1\rangle$ through the NOT gate, v_1 is transformed from $|0\rangle$ to $|1\rangle$ through the NOT gate. Through the second Toffoli gate, because the control bits v_0 and v_1 are both $|1\rangle$, the target bit a_{10} performs the Pauli-X gate (quantum non-gate) operation from $|0\rangle$ to $|1\rangle$. v_0 is transformed from $|1\rangle$ to $|0\rangle$ through the NOT gate, and v_1 is transformed from $|1\rangle$ to $|0\rangle$ through the NOT gate. This part is used to determine whether the colors of the two vertices v_0 and v_1 are the same. The output result of $|1\rangle$ indicates that the colors are equal, and the output result of $|0\rangle$ indicates that the colors are not equal.

Through the third Toffoli gate, because the control bit e_0 is $|0\rangle$ and a_{10} is $|1\rangle$, the target bit a_{20} remains unchanged as $|1\rangle$. This part is used to judge whether the two vertices just dyed meet the dyeing requirements. Here, because the edge e_0 input is $|0\rangle$, it means that the two vertices v_0 and v_1 are not adjacent. So the final output result is $|1\rangle$, which means that v_0 and v_1 are feasible to dye the same color.

Next, continue to judge other vertices according to the analysis just made. Through the fourth Toffoli gate, because the control bits v_0 and v_2 are both $|0\rangle$, the target bit a_{11} remains unchanged at $|0\rangle$; v_0 is transformed from $|0\rangle$ to $|1\rangle$ through the NOT gate, v_2 is transformed from $|0\rangle$ to $|1\rangle$ through the NOT gate. Through the fifth Toffoli gate, because the control bits v_0 and v_2 are both $|1\rangle$, the target bit a_{11} performs the Pauli-X gate (quantum non-gate) operation from $|0\rangle$ to $|1\rangle$. v_0 is transformed from $|1\rangle$ to $|0\rangle$ through the NOT gate, and v_2 is transformed from $|1\rangle$ to $|0\rangle$ through the NOT gate; Through the sixth Toffoli gate, because the control bit e_1 is $|0\rangle$ and a_{11} is $|1\rangle$, the target bit a_{20} remains unchanged as $|1\rangle$; Through the seventh Toffoli gate, because the control bits v_1 and v_2 are both $|0\rangle$, the target bit a_{12} remains unchanged at $|0\rangle$; v_1 is transformed from $|0\rangle$ to $|1\rangle$ through the NOT gate, v_2 is transformed from $|0\rangle$ to $|1\rangle$ through the NOT gate. Through the eighth Toffoli gate, because the control bits v_1 and v_2 are both $|1\rangle$, the target bit a_{12} performs the Pauli-X gate (quantum non-gate) operation from $|0\rangle$ to $|1\rangle$. v_1 is transformed from $|1\rangle$ to $|0\rangle$ through the NOT gate, and v_2 is transformed from $|1\rangle$ to $|0\rangle$ through the NOT gate; Through the ninth Toffoli gate, because the control bit e_0 is $|0\rangle$ and a_{12} is $|1\rangle$, the target bit a_{20} remains unchanged as $|1\rangle$.

Through the tenth MCT gate, because the control bits a_{20} , a_{21} , and a_{22} are all $|1\rangle$, the target bit o performs the Pauli-X gate (quantum non-gate) operation from $|0\rangle$ to $|1\rangle$. This part is used to merge the dyeing results. If the dyeing conditions of any two vertices meet the requirements (that is, the output result is $|1\rangle$), then the final output result is $|1\rangle$.

3.2 Quantum Circuit Decomposition

The output reversible circuit is a high-level description for the quantum algorithm of the graph coloring problem. It cannot be implemented directly on the quantum devices. It is well known that the Clifford + T gates constitutes a universal quantum gate set, and can be implemented by many quantum computers. We decompose the reversible circuits into equivalent Clifford + T circuits. Figure 5 shows part of the decomposed quantum circuits for the graph with 3 vertices. Table 5 shows the size and depth of the decomposed circuits for several reversible circuits in Table 4. We can see that the size and depth of the decomposed circuits grows rapidly, so a circuit optimization algorithm should be applied further.

Number of vertices	Number of gates					Depth	Number of qubits
	X	H	T	T'	CNOT		
3	15	20	36	27	68	186	14
4	30	38	72	54	108	310	24
5	50	62	120	90	180	514	37
6	75	92	180	135	270	769	53
7	105	128	252	189	378	1075	72

Table 5: The Number of Quantum Qubits and Gates.

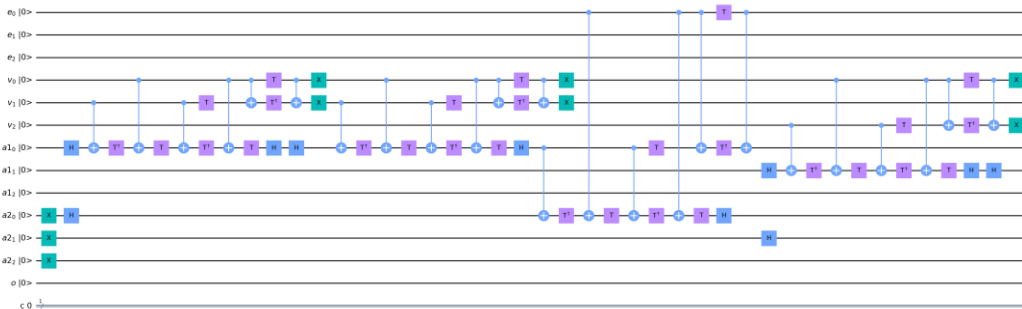


Figure 5: Part of the Decomposed Quantum Circuit When the Number of Vertices is 3.

4 CONCLUSIONS

In this paper, we give a classical algorithm to generate the quantum circuits for the 2-coloring problem of graphs. The algorithm can construct a high-level classical reversible circuit that check whether a 2-coloring strategy is correct or not. The reversible circuit uses the X gate, the Toffoli gate and MCT gates. Then we decompose the reversible circuits into equivalent Clifford + T circuits. For the future research direction, we will use the quantum circuit optimization algorithm to reduce the number of gates and depth for the resulting Clifford + T circuits.

Jinfeng He, <https://orcid.org/0009-0006-4641-2248>

Yao Zhang, <https://orcid.org/0009-0005-2147-2132>

Shanxian Lin, <https://orcid.org/0009-0003-1595-9492>

REFERENCES

- [1] Ananda, R.; Indra, Z.; Nasution, H.: Application of Graph Coloring on Nurse Work Scheduling at H. Adam Malik Hospital Medan Using the Tabu Search Algorithm, ZERO: Jurnal Sains, Matematika dan Terapan, 2022. <https://doi.org/10.30829/zero.v6i1.12451>
- [2] Appel, K.; Haken, W.: The Solution of the Four-Color-Map Problem, Scientific American, 237, 1977, 108–121. <https://doi.org/10.1038/scientificamerican1077-108>
- [3] Aslan, M.; Baykan, N.A.: A Performance Comparison of Graph Coloring Algorithms, International Journal of Intelligent Systems and Applications in Engineering, 4, 2016, 1–7. <https://doi.org/10.18201/ijisae.273053>

- [4] Bravyi, S.; Kliesch, A.; Koenig, R.; Tang, E.: Hybrid Quantum-Classical Algorithms for Approximate Graph Coloring, *Quantum*, 6, 2022, 678. <https://doi.org/10.22331/q-2022-03-30-678>.
- [5] Das, D.; Ahmad, S.A.; Kumar, V.: Deep Learning-Based Approximate Graph-Coloring Algorithm for Register Allocation, 2020 IEEE/ACM 6th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC) and Workshop on Hierarchical Parallelism for Exascale Computing (HiPar), 23–32, 2020. <https://doi.org/10.1109/LLVMHPChiPar51896.2020.00008>
- [6] DHondt, E.: Quantum Approaches to Graph Colouring, *Theoretical Computer Science*, 410, 2009, 302–309. Computational Paradigms from Nature. <https://doi.org/10.1016/j.tcs.2008.09.055>
- [7] Fabrikant, A.; Hogg, T.: Graph Coloring with Quantum Heuristics, in *Proceedings of the AAAI/IAAI*, 2002.
- [8] Fritsch, R.; Fritsch, G.: *The four-color theorem : History, Topological Foundations, and Idea of Proof*. Springer New York, NY, 1998. <https://doi.org/10.1007/978-1-4612-1720-6>
- [9] Gardahadi.: *Complexity Analysis of Basic Graph Coloring Algorithms*. 2018.
- [10] Goel, N.; Freericks, J.K.: Native Multiqubit Toffoli Gates on Ion Trap Quantum Computers, *Quantum Science & Technology*, 2021, 6. <https://doi.org/10.1088/2058-9565/ac1e02>
- [11] Hladký, J.; Král, D.; Schauz, U.: Brooks' Theorem via the Alon-Tarsi Theorem, *Discret. Math.*, 310, 2010, 3426–3428. <https://doi.org/10.1016/j.disc.2010.07.019>
- [12] Hu, W.; Yang, Y.; Xia, W.; Pi, J.; Huang, E.; Zhang, X.D.; Xu, H.: Performance of Superconducting Quantum Computing chips under different architecture designs, *Quantum Information Processing*, 2022, 21. <https://doi.org/10.1007/s1128-022-03571-0>.
- [13] Krlev, V.S.; Krleva, R.: An Analysis Between Different Algorithms for the Graph Vertex Coloring Problem, *International Journal of Electrical and Computer Engineering*, 2023. <https://doi.org/10.11591/ijece.v13i3.pp2972-2980>
- [14] Langberg, M.; Chekuri, C.: Graph Coloring, In *Proceedings of the Encyclopedia of Algorithms*, 2008. https://doi.org/10.1007/978-0-387-30162-4_170
- [15] Lewis, R.R.: *A Guide to Graph Colouring: Algorithms and Applications*, 1st ed.; Springer Publishing Company, Incorporated, 2015.
- [16] Li, Z.; Zhang, W.; Zhang, G.; Dai, J.; Hu, J.; Perkowski, M.A.; Song, X.: An Extended Approach for Generating Unitary Matrices for Quantum Circuits, *Cmc-computers Materials & Continua*, 62, 2020, 1413–1421. <https://doi.org/10.32604/cmc.2020.07483>
- [17] Malkawi, M.I.; Hassan, M.A.H.; Hassan, O.A.H.: A New Exam Scheduling Algorithm Using Graph Coloring, *Int. Arab J. Inf. Technol*, 5, 2008, 80–86.
- [18] Mitarai, K.; Negoro, M.; Kitagawa, M.; Fujii, K.: Quantum Circuit Learning, *Physical Review A*, 2018. <https://doi.org/10.1103/PhysRevA.98.032309>
- [19] Ouerfelli, L.; Bouziri, H.: Greedy Algorithms for Dynamic Graph Coloring, 2011 International Conference on Communications, Computing and Control Applications (CCCA), 2011, 1–5. <https://doi.org/10.1109/CCCA.2011.6031437>
- [20] Do, M.; Wang, Z.; O’Gorman, B.; Venturelli, D.; Rieffel, E.G.; Frank, J.: Planning for Compilation of a Quantum Algorithm for Graph Coloring. In *Proceedings of the ECAI 2020 - 24th European Conference on Artificial Intelligence*, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020); Giacomo, G.D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; Lang, J., Eds. IOS Press, *Frontiers in Artificial Intelligence and Applications*, 325, 2020, 2338–2345.
- [21] Raychev, N.; Chuang, I.L.: *Quantum computation and quantum information*. 2010.
- [22] Samarasekara, W. An Application of Graph Coloring Model to Course Timetabling Problem, 2019.

- [23] Shimizu, K.; Mori, R.: Exponential-Time Quantum Algorithms for Graph Coloring Problem, *Algorithmica*, 84, 2022, 36033621. <https://doi.org/10.1007/s00453-022-00976-2>
- [24] Sipayung, T.N.; Suwilo, S.; Gultom, P.; Mardiningsih.: Implementation of the greedy algorithm on graph coloring, *Journal of Physics: Conference Series*, 2022, 2157. <https://doi.org/10.1088/1742-6596/2157/1/012003>
- [25] Tabi, Z.; El-Safty, K.H.; Kallus, Z.; Haga, P.; Kozsik, T.; Glos, A.; Zimboras, Z. Quantum Optimization for the Graph Coloring Problem with Space-Efficient Embedding, In *Proceedings of the 2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2020, 56–62. <https://doi.org/10.1109/QCE49297.2020.00018>.
- [26] Thapliyal, H.; Munoz-Coreas, E.: Design of Quantum Computing Circuits, *IT Professional* 21, 2019, 22–26. <https://doi.org/10.1109/MITP.2019.2943134>
- [27] Zajtka, M.: A short proof of Brooks' theorem. 2018.