Circuit Design for $k$-coloring Problem and Its Implementation on Near-term Quantum Devices

Amit Saha$^{1,2}$, Debasri Saha$^1$, and Amlan Chakrabarti$^1$
$^1$A. K. Choudhury School of Information Technology, University of Calcutta, Kolkata - 700 106, India
$^2$ATOS, Pune, India

Abstract—Nowadays in Quantum Computing, the implementation of quantum algorithm has created a stir since Noisy Intermediate-Scale Quantum (NISQ) devices are out in the market. Researchers are mostly interested in solving NP-complete problems with the help of quantum algorithms for its speed-up. As per the work on computational complexity by Karp [1], if any of the NP-complete problem can be solved then any other NP-complete problem can be reduced to that problem in polynomial time. In this paper, $k$-coloring problem (NP-complete problem) has been considered to solve using Grover’s search. A comparator-based approach has been used to implement $k$-coloring problem which enables the reduction of the qubit cost compared to the state-of-the-art. An end-to-end automated framework has been proposed to implement the $k$-coloring problem for any unweighted and undirected graph on any available Noisy Intermediate-Scale Quantum (NISQ) devices, which helps in generalizing our approach.

Index Terms—$k$-coloring problem, Grover’s Search, NISQ.

I. INTRODUCTION

As the development of Noisy Intermediate-Scale Quantum (NISQ) computer [2] has achieved a remarkable success in recent times, everyone has shown a striking interest to implement quantum algorithms, which give a potential speedup over their classical counterparts. With the growing quantum wave, there is a huge urge for implementing NP-complete problems on near term quantum devices. It would be helpful for any naive person, if we could provide them with an automated end-to-end framework for implementing an NP-complete problem so that they can easily map their computational problem without having much knowledge about gate-based quantum circuit implementation. In this paper, we have focused on $k$-coloring problem.

The $k$-coloring problem finds whether a given graph’s vertices or nodes are properly colored or not using $k$ colors by taking into account that every two vertices linked by an edge have different colors. Suppose $n$ is the number of nodes of a given graph, $k$ is the number of colors, then to find the exact solution using a classical algorithm requires $O(2^{n\times \log k})$ number of steps. Whereas, using the decision oracle and the diffusion operator of Grover’s algorithm [3], finding the exact solution requires $O(\sqrt{N})$ number of iterations where $N$ is $2^{n\times \log k}$. Previously in [4] [5], Graph coloring problem using Grover’s algorithm has been discussed in the context of quantum system. But, in [6] SAT reduction technique has been used to solve 3-coloring problem and gave an end-to-end framework for implementing it in the IBMQ quantum processor [7]. For this SAT reduction technique, the qubit cost is immense, hence circuit cost becomes inefficient.

In this paper, we have proposed an automated qubit cost-efficient comparator-based approach to implement $k$-coloring problem for mapping high level description to any hardware-specific low-level quantum operations with an abstraction. The novelty of this paper is as follows:

- We propose an end-to-end automated framework for $k$-coloring problem using quantum search algorithm, which takes graph and number of color ($k$) as input and automatically implements on the NISQ device.
- We propose a comparator-based approach to implement the $k$-coloring problem which has less qubit cost comparing to the state-of-the-art.
- The framework is designed in such a way that the Quantum solution of $k$-coloring problem can be mapped into any available NISQ devices, which makes our approach generalized in nature.

The structure of this paper is as follows. The synopsis of Grover’s algorithm, Quantum circuits, and NISQ devices are described in Section II. In Section III, the proposed methodology has been discussed. The implementation of $k$-coloring problem has been illustrated in Section IV. Concluding remarks appear in Section V.

II. BACKGROUND

In this section, we have mainly described about quantum circuit, Grover’s algorithm and finally NISQ devices.

A. Quantum circuit

Any quantum algorithm can be expressed or visualized in the form of a quantum circuit. These quantum circuits constitute of logical qubits and quantum gates [8].

1) Qubits: Logical qubit that is used to encode input or output of a quantum algorithm is known as data qubit. There is another type of qubit that is used to store temporary results are known as ancilla qubit.  

2) Quantum Gates: Unitary quantum gates need to be applied on qubits to modify the quantum state of a quantum algorithm. To synthesize our proposed circuit, we use NOT gate, Controlled-NOT gate, Toffoli gate, Hadamard gate and Multi Control Toffoli gate (MCT). All the mentioned gates except the MCT are described in Table I. The description of MCT gate is as follows:

| Gate | Operation |
|------|-----------|
| NOT  | $|x\rangle \rightarrow |\bar{x}\rangle$ |
| CNOT | $|x,y\rangle \rightarrow |x,y\oplus x\rangle$ |
| Toffoli | $|x,y,z\rangle \rightarrow |x,y,z\oplus x\lor y\rangle$ |
| Hadamard | $|x\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ |
| MCT | $|x,y,z\rangle \rightarrow |x,y,z\oplus x\lor y\lor z\rangle$ |
Multi-Controlled Toffoli Gate: There are n number of inputs and outputs in an n-bit MCT. This MCT gate passes the first n − 1 inputs, which are referred to as control bits to the output unaltered. It inverts the nth input, which is referred to as the target bit if the first n − 1 inputs are all ones. An MCT gate is shown in Figure 1. Black dots • represent the control bits and the target bit is denoted by a ⊕.

B. Grover’s Algorithm

Grover’s algorithm has two parts, namely oracle and diffusion operator. The oracle depends on the specific instance of the search problem. The diffusion operator block is also known as inversion over the average operator and it amplifies the amplitude of the marked state to increase its measurement probability. The block diagram of a typical Grover’s algorithm is shown in Figure 2.

Sign Flip: Flip the sign of the vectors for which $U_f$ gives output 1.

Amplitude Amplification: We need to perform the inversion about the average of all coefficient of the quantum state for a certain number of iterations to get the coefficient of the marked state is large enough that it can be obtained from a measurement with probability close to 1. This phenomenon is known as amplitude amplification which is performed by using diffusion operator.

Number of Iterations: Grover’s Search algorithm requires $\sqrt{N/M}$ many iterations to get the probability of one of the marked states $M$ out of total $N$ number of states set.

C. NISQ Devices

NISQ devices are “noisy,” due to the constraint of the number of qubits, hence one has to allow a certain range of error while estimating the simulated result of a quantum state [2]. Superconducting quantum circuits, ion trap, quantum dot, neutral atom are the most popular NISQ technologies to implement the quantum circuit. Every one of them has a specific qubit topology, as shown in Figure 3 so as to map the logical synthesized circuit to quantum hardware. Table II illustrates certain 1-qubit and 2-qubit gates that are supported by most of the quantum hardware. One has to realize their logical quantum gates to these hardware-specific gates to make it hardware compatible for implementation.

### III. Proposed Methodology of Circuit Synthesis for $k$-Coloring Problem using Grover’s Algorithm

The flowchart as shown in Figure 4 describes the complete flow of our proposed automated end-to-end framework. Our framework is mainly based on three algorithms: AutoGenOracle_K-color, MCT_Realization and SABRE(Qubit Mapping). Firstly, adjacency matrix of the given graph and the number of color ($k$) is given as input to AutoGenOracle_K-color algorithm and we get the quantum circuit netlist in the form QASM as output. AutoGenOracle_K-color algorithm automatically generates Oracle circuit for $k$-coloring problem using Grover search and is based on the newly designed comparator. Now, MCT_Realization algorithm takes generated circuit netlist as input and realizes MCT gates to NISQ hardware compatible 1-qubit and 2-qubit gates [10]. Finally SABRE [11] algorithm has been used for mapping generated circuit by NISQ devices based on the qubit topology.

This section outlines the proposed methodology for the Oracle circuit synthesis of the $k$-Coloring problem as an application of the Grover’s search algorithm.

### TABLE I

| Quantum Gates     | Matrix Representation | Circuit Representation |
|-------------------|-----------------------|------------------------|
| Hadamard Gate     | $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ | $\begin{pmatrix} | \psi\rangle \\ \phi\rangle \end{pmatrix}$ |
| Not Gate          | $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ | $\begin{pmatrix} | \psi\rangle \\ \phi\rangle \end{pmatrix}$ |
| Controlled-NOT Gate | $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ | |
| Toffoli Gate      | $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ |
A. Proposed Oracle for k-Coloring Problem

The quantum circuit block of Oracle for the k-coloring problem is shown in Figure 7. The construction of Oracle for k-coloring problem is divided into five parts starting with initialization, which is essentially required in Grover’s Algorithm.

1) Initialization: If there are \( n \) vertices, \( e \) edges in the input graph and \( k \) is the number of colors, then the total number of data qubits required to represent all the colored vertices are \( n \times \lceil \log_2 k \rceil \). The Oracle checks for all the right combination of properly colored vertices with \( k \) or fewer colors from a combination of all possible colored vertices. Hence, a superposition of \( m = n \times \lceil \log_2 k \rceil \) qubits will generate all possible combination of colored vertices. The initial data qubits in Figure 7 include \( m \) qubits prepared in the ground state \( |\psi\rangle = |0\rangle^{\otimes m} \), due to the re-usability property of ancilla qubits, \( r = n \) ancilla qubits in the exited state \( |\theta\rangle = |1\rangle^{\otimes r} \) (These \( r \) ancilla qubits are required to prepare Invalid Color detector block and Comparator block which are described in next subsection thoroughly), one ancilla qubit in the ground state \( |\zeta\rangle = |0\rangle \) (1 ancilla is required iff invalid color exists), and one output qubit in the exited state \( |\phi\rangle = |1\rangle \) is required to perform CNOT/Toffoli/MCT operation of the Oracle. This entire initialization can be mathematically written as:

\[
|\psi\rangle \otimes |\theta\rangle \otimes |\zeta\rangle \otimes |\phi\rangle = |0\rangle^{\otimes m} \otimes |1\rangle^{\otimes r} \otimes |0\rangle \otimes |1\rangle
\]

2) Hadamard Transformation: After the initialization, the Hadamard transform \( H^{\otimes m} \) on data qubits and \( H \) on output qubit is performed, therefore all possible states are superposed as \( |\psi_0\rangle \otimes |\theta_0\rangle \otimes |\zeta_0\rangle \otimes |\phi_0\rangle \), where

\[
|\psi_0\rangle = \frac{1}{\sqrt{2^m}} \sum_{i=0}^{2^m-1} |i\rangle
\]

\[
|\theta_0\rangle = |1111.....r\times\text{(times)}\rangle
\]

\[
|\zeta_0\rangle = |0\rangle
\]

\[
|\phi_0\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)
\]

3) Proposed \( U_f \) Transformation:: This proposed unitary \( U_f \) transformation has two distinct parts.

1)Reduction of Invalid Colors: Since \( c = \lceil \log_2 k \rceil \), hence we consider maximum \( 2^c \) colors. If \( 2^c = k \), then all colors are valid colors, else there will be a set of \( 2^c - k \) invalid colors. The search space should be optimized with valid colors. This can be carried out using the following steps:

Qubit Activation: Colors are needed to be numbered as \( \{0, 1, 2,..., 2^c - 1\} \). After the Hadamard transformation, the input data qubit lines act as the binary representation of combination of all possible colored vertices. But, the oracle checks only the combination of valid colors \( k \). To make sure that the Oracle is checking only the \( k \)-colored combination of vertices, all the input qubit lines are needed to be in the excited state \( |1\rangle \) for those particular combinations of invalid colors by making input qubit lines suitable as control lines for CNOT/Toffoli/MCT operation. A number of NOT gates have to be imposed on the input qubit lines, which are in the ground state \( |0\rangle \) followed by the application of 'Invalid Color Detector'. This 'Qubit Activation' has to be applied again after 'Invalid Color Detector' to return back to the initial superposed quantum state.

Invalid Color Detector: If any invalid color is detected in any combination of colored vertices then that combination is discarded using the following function ICD (Invalid Color Detector):

\[
ICD(I_1, I_2, ..., I_n, f) = \begin{cases} f = 0, & \text{if } I_1 \text{or } I_2 \text{or } ... \text{or } I_n = \text{Invalid color} \vspace{1mm} \\ f = 1, & \text{No invalid color.} \end{cases}
\]  

Figure 5 describes the circuit synthesis of 'Invalid Color Detector’ for \( n \) vertices, where \( I_1, I_2, ..., I_n \) are the data qubits.

(2) Binary Comparator: A newly proposed binary comparator circuit can be defined as:

\[
\text{Comparator}(a, b, f) = \begin{cases} f = 0, & \text{if } a = b \vspace{1mm} \\ f = 1, & a \neq b. \end{cases}
\]
where \( a \) and \( b \) are the comparing inputs which represent the colored vertices of the given graph and \( f \) is the ancilla qubit. Circuit synthesis for 2-qubit and 4-qubit comparator is shown in Figure 6. CNOT, NOT, Toffoli/MCT gates are used to design the complete circuit synthesis for the binary comparator.

![Figure 6. Example Comparator: (a) 2-qubit; (b) 4-qubit](image)

With the help of these invalid color reduction function and newly proposed comparator, The design of \( U_f \) of an Oracle for \( k \)-coloring problem is effectively developed.

4) MCT Operation: The output qubit state \( |\phi_0\rangle \) is initially set as \( \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) \). Applying an MCT gate on the output line considering ancilla qubits as control, results in an eigenvalue kickback, \(-1\), which causes a phase shift for the respective input state/states, which helps to find out all the combination of properly colored set of vertices. The algorithm that generates the gate level synthesis of the proposed method is outlined in the next subsection.

B. Proposed Algorithm for Oracle Circuit Synthesis

The proposed algorithm Algorithm 1 (AutoGenOracleK-Coloring) of automated oracular circuit synthesis for the \( k \)-coloring problem is illustrated in this subsection. The algorithm takes as input the adjacency matrix of the given graph and the number of colors \( k \). The output of the algorithm is a circuit netlist in the form of QASM.

From the details of the adjacency matrix and the number of given color, it can be easily estimated that the total number of qubit lines required to generate the Oracle circuit. All the input data qubits are initialized with \( |0\rangle \) followed by Hadamard, ancilla lines \( (A_r) \) are initialized with \( |1\rangle \), ancilla line \( A_{r+1} \) is initialized with \( |0\rangle \) and the output line is initialized with \( |1\rangle \) followed by Hadamard. First of all, apply Invalid Color Detector with suitable Qubit Activation (if invalid color exists) with \( I_r, A_r \) as control and \( A_{r+1} \) as the target. Then, between two adjacent vertices \((i,j)\), a comparator circuit is used with two input lines \((i,j)\) as control and the ancilla line \((A_r)\) as output and perform this same task for all the adjacent vertices. Then, an MCT gate is used with all the ancilla lines \( A_r \) and \( A_{r+1} \) as control and the output line as output for the flip operation of Grover’s Oracle. To mirror everything of the Oracle circuit, we have repeated the previous steps as shown in Algorithm 1.

C. Circuit Cost Estimation

The design of generalized Oracle for our algorithm is already described. Now, the circuit cost analysis of the oracular circuit is given in Table 11.

For \( n \)-vertices graph and \( k \) given color, \( n \times \lceil \log_2 k \rceil \) data qubits are required. For \( n \)-vertices graph, at most \( n+1 \) number of ancilla are needed and at most \( O(n^2 \times \log_2 n) \) gates are required to design the oracular circuit. The gate-optimized circuit synthesis of the 3-coloring problem for example graph

```
ALGORITHM 1: AutoGenOracleK-Coloring\((G(V, E))\)
1: INPUT : Adjacency matrix \( \text{adj}(n, n) \) of graph \( G(V, E), V = n \) and \( E = e \) where, \( V \) is the set of nodes and \( E \) is the set of edges, Number of input data qubit lines required \( I_r = n \times \lceil \log_2 k \rceil \) (input lines for \( n \) nodes and \( k \) colors)+ ancilla lines required= \( n \) +1 ancilla line for reduction of invalid colors(if required) +1(output line \( O \)), \( A_r \) represents ancilla line where, \( 1 \leq r \leq n \), \( A_{r+1} \) represents ancilla line for invalid color (if required).
2: OUTPUT : Circuit netlist (QASM)
3: Initialize \( I_r \) input lines with \(|0\rangle \) followed by Hadamard gate, ancilla lines \( A_r \) with \(|1\rangle \), \( A_{r+1} \) with \(|0\rangle \), and output line \( O \) with \(|1\rangle \) followed by a Hadamard gate.
4: Apply Invalid Color Detector (if required) for all possible invalid colors with suitable Qubit Activation with \( I_r, A_r \) as control and \( A_{r+1} \) as target.
5: \( l \leftarrow n, f \leftarrow 1 \)
6: for \( i \leftarrow 1 \) to \( n - 1 \) do
7: \( r \leftarrow f, m \leftarrow f \)
8: for \( j \leftarrow i + 1 \) to \( n \) do
9: if \( \text{adj}(i, j) \leftarrow 1 \) \((i \text{ and } j \text{ are connected by an edge}(e))\) then
10: Use a comparator circuit with the input lines \( (I_i, I_j) \) corresponding \((i, j)\) as control and the ancilla line \( A_r \) as target.
11: \( r \leftarrow r + 1 \)
12: end if
13: end for
14: if \( r > f + 1 \) then
15: Use a Toffoli/MCT gate with all ancilla lines \( A_r \) as control and \( A_{r+1} \) as target
16: \( l \leftarrow l - 1 \)
17: for \( m \leftarrow i + 1 \) to \( n \) do
18: if \( \text{adj}(i, j) \leftarrow 1 \) \((i \text{ and } j \text{ are connected by an edge}(e))\) then
19: Use a comparator circuit with the input lines \( (I_i, I_m) \) corresponding \((i, j)\) as control and the ancilla line \( A_m \) as target.
20: \( m \leftarrow m + 1 \)
21: end if
22: end for
23: else if \( r = f + 1 \) then
24: \( f \leftarrow f + 1 \)
25: end if
26: end for
27: Use an MCT gate with all the ancilla lines \( A_1, A_2, \ldots A_{r+1} \) as control and \( O \) as output.
28: Repeat step 5-26.
29: Repeat step 4.
```
of three vertices with three connected edges ($K_3$) is shown in Figure 8.

**D. Diffusion**

The second part of Grover’s algorithm is the circuit implementing the function of diffusion. When the operation is applied to a superposition state, it actually keeps the component in the $|\psi_0\rangle$ direction unchanged, while inverting the components in dimensions that are perpendicular to $|\psi_0\rangle$. This can be represented as

$$I|\psi_{\perp 0}\rangle = -I|\psi_0\rangle$$

where,

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$$

The diffusion operator is a unitary matrix. The general matrix for the diffusion operator for a $d$-dimensional quantum system is shown below:

$$diff_d = \begin{pmatrix} 2^d - 1 & 2^d - 1 & \cdots & 2^d - 1 \\ 2^d - 1 & 2^d - 1 & \cdots & 2^d - 1 \\ \vdots & \vdots & \ddots & \vdots \\ 2^d - 1 & 2^d - 1 & \cdots & 2^d - 1 \end{pmatrix}$$

**IV. MAPPING OF $k$-COLORING PROBLEM TO NISQ DEVICES**

This section focuses on the mapping of generated Oracle circuit to NISQ devices through MCT realization and SABRE algorithm for qubit mapping.

**A. Realization of MCT Gate**

Figure 9 shows how to decompose MCT gate to NISQ compatible 1-qubit and 2-qubit gates [10]. Firstly MCT gate needs to be decomposed to MCZ gate. Then, the realization of MCZ gate into $MCZ(R_{0\pi(*)})$ is performed. Lastly, $MCZ(R_{0\pi(*)})$ is reduced to 1-qubit and 2-qubit gates without using any ancilla qubit.

**B. Qubit Mapping to NISQ Devices**

Since, our proposed quantum circuit is logical, hence there is no constraint of qubit connectivity. For NISQ devices, there exists a specific qubit topology or coupling graph. Coupling graph defines the interaction between two physical qubits. This varies for different NISQ devices. Thus, it is obvious that mapping the logical circuit to the physical one is a challenge. The solution to this problem is the insertion of SWAP gates between the two qubits to satisfy the hardware constraint without compromising on the logic of the quantum circuit. The idea of a good qubit mapping problem is to minimize the number of SWAP insertion gates and minimize the depth of the circuit. Li et. al. proposed SWAP-based BiDiREctional heuristic search algorithm (SABRE) in [11], which is a benchmark, since it deals with any arbitrary qubit topology for any NISQ device. Mainly three features make SABRE stand out. Firstly, it doesn’t perform an exhaustive search on the entire circuit, but it performs a SWAP-based heuristic search considering the qubit dependency. It then optimizes the initial mapping using a novel reverse traversal technique. Last but not the least, the introduction of the decay effect for enabling the trade-off between the depth and the number of gates of the entire algorithm. We use SABRE protocol so that our proposed circuit can easily be mapped to any arbitrary qubit topology.

**C. Experimental result of $k$-coloring Problem in NISQ Device**

As shown in Figure 8, the generated oracle circuit for example graph has been taken as an example case for the
simulation of $k$-coloring problem which is performed on IBMQ cloud based physical device [7].