ISSN No:-2456-2165

Solving the General Planar Graph Coloring Problem Using Grover's Algorithm

Lenish Pandey[1]; Swayam Chaulagain[1]; Madhav Khanal[1]; Pratyush Bhattrai[1,2]
Sampada Wagle[1,] Sooraj Sahani[1]
[1]Incubate Nepal, Kathmandu, Nepal
[2]Department of Physics, University of Bristol, UK

Abstract: In this paper, we discuss the structure and implementations of the planar graph coloring problem (PGCP). We briefly look at well-known classical algorithms used to solve the PGCP but primarily focus on the quantum computational angle. Grover's search is a well-known quantum algorithm that offers a quadratic advantage relative to its classical counterparts. Traditionally, this algorithm is used for search and sorting; in this paper, we inspect its application to the PGCP and build a corresponding quantum circuit. While we take a specific case of specialists in Nepalese hospitals and optimize their placements, the approach we have shown in this paper has wide implications for efficiently solving a huge range of optimization problems in health, transport, and so on.

I. INTRODUCTION

Mathematical and computational methods of problem-solving have grown exponentially over the past century, providing efficient and effective solutions to various problems. From the classic brute force and manual calculations, these methods have evolved to yield faster calculations with efficient algorithms. Mathematical algorithms have been integrated into different fields of human endeavors, such as finance, health, agriculture, education, and so on. The applications are open to exploration in many endeavours. The development of problem-oriented computational algorithms dates back to the mid-20th century (Knuth, 1977). Researchers including Turing (1936), Dantzig (1951), Hoare (1961), Haigh (1993), Copeland (2004), Belvos (2013), Montanaro (2016), Childs et. al. (2018) have reported compelling algorithms ranging from logic, linear functions, universal computation to optimizations. With the advancement of quantum mechanics, quantum computing algorithms have also been intensively used in the fields of optimization, cryptography and cryptoanalysis. These quantum algorithms offer a speedup in calculation due to carrying out multiple calculations simultaneously.

Taking Nepal as a sample location to map out dif ferent real life scenarios in terms of mathematical and computational models for efficient problem-solving, a number of areas can be considered. We can map out the location of disaster-prone areas and based on the distance between the major necessities, allocate the appropriate resources. We Can effectively plan hydropower plant schedul ing, based on the energy consumption of a certain area, the number of workers, total electricity production, as well as the medium of transmission. We can optimize bus routing based on distance and traffic mobilization for deterministic time frames. We can also allocate network bandwidth in a way that meets transmission requirements with the least interference and maximizes network efficiency.

The aforementioned issues, based on the problem domains and requirements, fall under resource allocation and effective scheduling. For a general resource allocation problem, a set of resources is to be allocated to different specifications. Different constraints are then defined which adds objectivity to the problem. Additional constraints can be defined to maximize the output function based on the problems. Likewise, as a scheduling problem goes, a set of works to be scheduled for a given time based on work-specific and time-specific parameters can be determined.

For our paper, the problem we shall be focusing on is to allocate specialist doctors to different hospitals in the Kathmandu Valley such that no doctor of similar expertise lands in the same or even adjacent hospitals at a given period of time. The hospitals have been chosen based on accessibility, proximity, clinical metrics, and resources. This sort of problem with adjacency in terms of resource allocation and scheduling can be modeled using graph coloring.

Various classical algorithms have been developed to address the graph coloring problem. However, we have opted for the use of the quantum algorithm "Grover's algorithm" to solve the problem. After mapping the information of hospitals and specialists in the graph coloring problem, we apply Grover's algorithm which, through re peated iterations, gives us the most effective solution. The sections to follow cover the details of mapping a graph coloring problem using different classical algorithms as well as implementing Grover's algorithm.

- *Problem Introduction*

Leading Nepali news portals have often reported problems in health services due to the unavailability of special doctors in rural Nepal. A fairly recent account of the threatened future of Nepal's health sector due to a lack of superspecialist doctors has been reported (OnlineKhabar English News, 2023). According to the Nepal Medical Council (NMC), there are 10,080 specialist doctors as of January 2023. The specialist doctors have not been mobilized properly with all the specialists being concentrated in bigger cities, and due to the very limited number of specialists, it is very problematic for everyone to get access to effective health services. Thus, we tried to generate an algorithm that would allocate the specialists to different hospitals based on pre-set constraints, which in our case is proximity.

For our solution sampling, a total of 8 hospitals were taken, and based on clinical facilities available and the distance, edges were defined. The hospitals under observation are Norvic International Hospital, Civil Hospital, B&B Hospital, Nepal Mediciti Hospital, Megha Hospital, Patan Hospital, Sumeru City Hospital and Star Hospital. Based on the size and distance between the hospitals, we generated a map which would encompass the necessary information of the hospitals (vertices), and the edges. For further simplicity, we take the following graph assigning a numerical vertex to the hospitals.

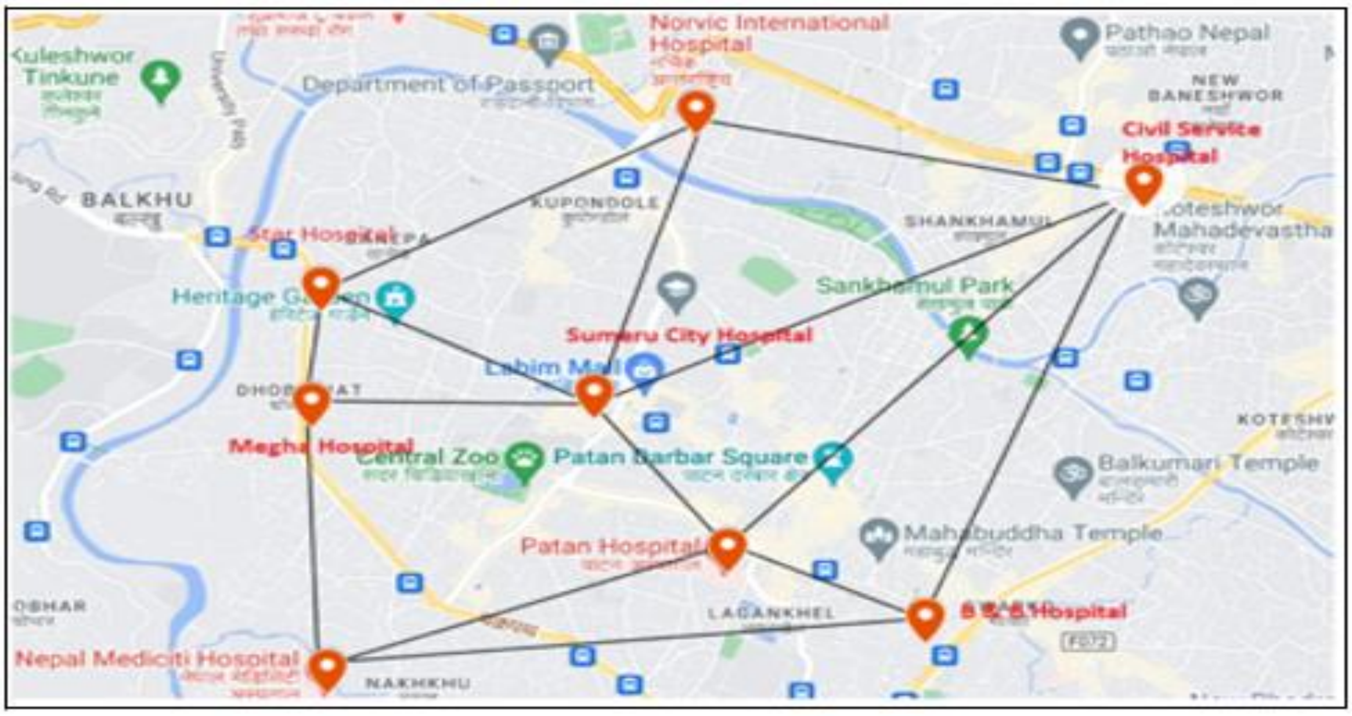

Fig 1 Map of Kathmandu Valley with Hospitals Marked as Nodes.

Based on the above graph, we can know the ad jacent nodes. Let E = (0,1), (0,5), (0,6), (0,7), (1,2), (1,6),(2,3), (2,6), (3,4), (3,6), (4,5), (4,7), (5,7), (6,7)be the set of all the edges in the graph. To solve the graph coloring problem, any two vertices are associated with an edge. i.e. two adjacent vertices should not have the same color. In our case, the color represents the specialist doctors.

II. GRAPH COLORING PROBLEM

A graph is a collection of vertices(nodes) connected by the edges. Typically, vertices of graphs are represented by names or properties. Edge is often used to link any two vertices of the graph. In terms of symbols, we represent graphs as G, vertices as V, and edges as E. The vertices having an edge between them are often called adjacent vertices. Graphs are either directed or undirected. Edges of directed graphs have direction associated with them while the edges of undirected graphs don't have any direction. All the graphs discussed in this paper will be undirected graphs. Undirected graphs in our paper are simple graphs, that is there won't be more than one edge connecting the same pair of vertices.

Graph coloring, just like its name, is a way of coloring vertices of the graph such that no two adjacent vertices share the same color. For this kind of coloring the easiest hack is to use different colors for each node. Since the total number of nodes in a graph doesn't have any restrictions, using different colors for different nodes is not feasible. Thus, for proper coloring of a graph, one should color the graph using a minimum number of colors.

The lowest number of colors required to color a graph(G) is called the chromatic number of G (Boyer et al., 1998). A graph G with a chromatic number (G)=k is k chromatic. Graph G whose vertices can be colored using k colors is called k colorable (Boyer et al., 1998). Normally there are three types of graph coloring: Vertex coloring, edge coloring, and face coloring (Diao, 2010). In this paper, we will stick with vertex coloring. Discussing further the chromatic number of a graph, we will discuss one of the landmark achievements in the field of graph theory widely known as the Four Coloring Theorem. The four-coloring theorem implies that for any planar graphs, their chromatic number is at most four. In other words, we can always color a planar graph with 4 colors.

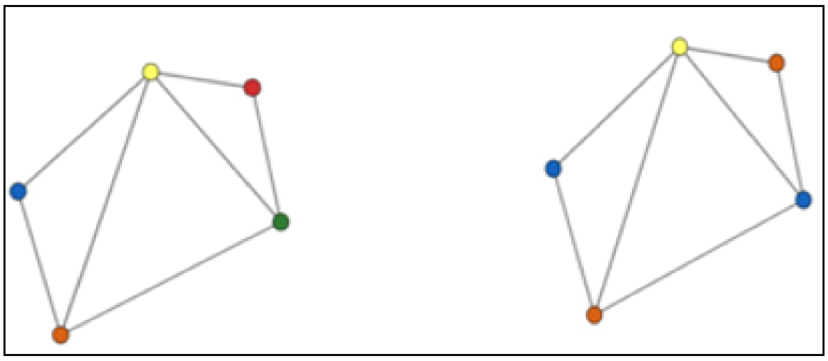
Fig 2 Planar Graphs with 5 Vertices and 7 Edges.

This interesting conjecture was first conjectured by Francis Guthrie in 1852 and remained unsolved for more than a century. Finally, the major proof was given in 1977 by Appel and Haken (K. Appel & W. Haken, 1977). Their proof was largely computer-based as it required solving too many cases. Whether this kind of computerized proof actually constituted proof in the mathematical community is still controversial.

Note that this theorem is limited to planar graphs. Because of this interesting boundary on chromatic numbers for planar graphs, we will be dealing with planar graphs in our paper. So let's look at what exactly are planar graphs.

Basically, a planar graph is a graph that can be drawn in the plane such that no two edges cross except at a vertex. But we can't ensure if a graph is planar just by looking at it. You can see Fig 2 as an illustration.

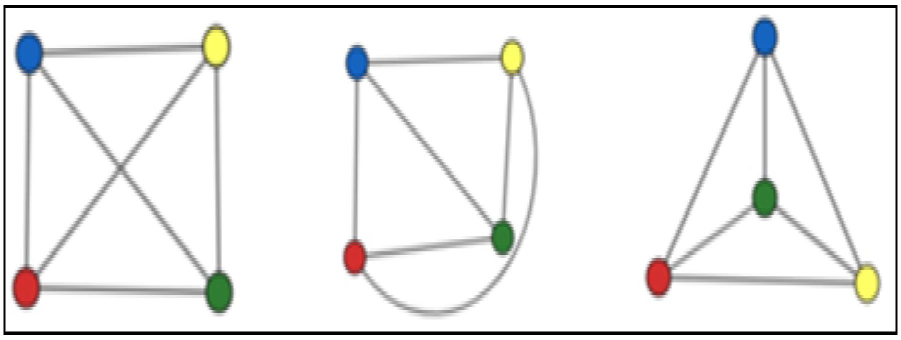
Fig 3 Different Kinds of Planar Graphs with 4 Vertices.

To overcome this problem, Euler formulated a famous theorem known as Euler's theorem. The theorem states that for any planar graph, No. of Vertices(v) Number of Edges(e) + regions(r) equals 2, i.e. v e+r=2. The number 2 in this theorem is not random as you can notice 2 usually has something to do with the plane. In this theorem, the region(r) of a planar graph is basically sections of a flat surface separated by a planar graph. You can look at Figure 2 for its illustration. Imagine erasing vertices from the surface, it breaks into separate pieces, and each piece is called a region. We also need to be aware that there's always one special outer region that contains all the parts of the surface that go on forever. For a region, degree (deg) is the number of edges that are adjacent to the region, written as deg(R). As an example of Euler's theorem, you can see it works in the graph below. Since, v=4, e=6 and r=4, 4 6+4=2.

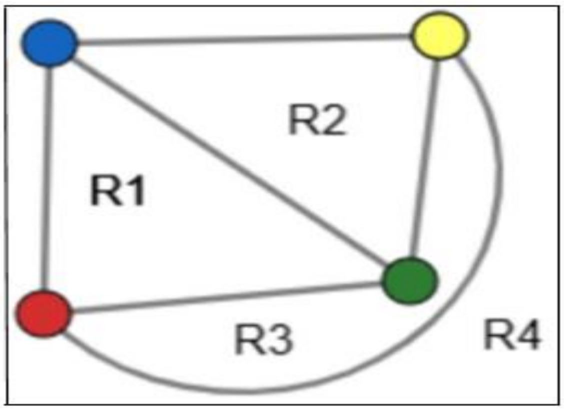

Fig 4 A Planar Graph to Illustrate Euler's Theorem

Euler's theorem can be proved using simple mathematical induction (Berman & Williams, 2009). Euler's theorem itself doesn't help us much to see if the graph is planar since we need to redraw the graph, but some corollary of this theorem can be very useful for us to test the pla narity of a graph. If G is a planar graph with vertices (v) and edges (e), with $v \geq 3$, then it must satisfy the inequality $e \leq 3v - 6$.

- *Proof*:

We can notice that, for each region of a planar graph, its degree is 3, and each edge is adjacent to two regions. Now, we can derive that $2e$ =3r degrees of r regions of a graph. Thus, $2e = 3r$ Substituting this in Euler's theorem, we can prove the inequality. However, this theorem is not two sided so we need to be careful. For all the planar graphs this inequality must be satisfied, but a graph satisfying this inequality doesn't imply that it is planar. For example: In figure 5 you can see that v=5 and e=10. So, 10 is not 9, the graph is not planar. Whereas in Figure 5, v=6 and e=9 and 9. If we check, the graph is not planar.

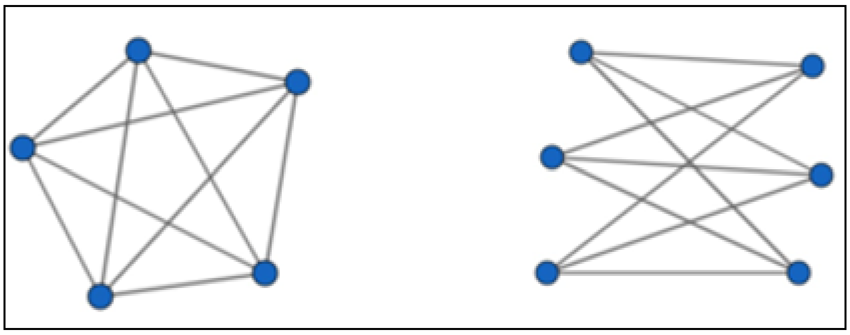
Fig 5 Planar and Non-Planar Graphs

➢ *Why Study Graph Coloring Problems?*

Graph coloring problem is one of the most important aspects of graph theory because of its real-life implications like scheduling, resource allocation, assigning radio frequency, map coloring, and many others. However, to date, there is no efficient algorithm for solving the graph coloring problem. It is one of the well-known Np complete problems. Np complete problems are the problems which don't have any established polynomial time algorithm. Polynomial-time algorithms are considered to be efficient because the execution times do not grow rapidly as the problem size increases, unlike exponential-time algorithms. Thus finding a polynomial time algorithm for any of the np complete problems can solve all of them.

➢ *Classical Solutions*

One of the popular classical algorithms for solving graph coloring problems is the Backtracking Algorithm. Backtracking is a classical approach to solving graph coloring using recursion. It is proven better than other classical and brute force methods, because the paths leading to false solutions are terminated earlier, preventing any further branches on that path. Although the upper bound of time complexity for both the backtracking and brute force methods are of $O(m^n)$.

Here, m is the number of colors and n is the number of nodes, the average time complexity of backtracking is lesser. Here is how backtracking works:

We want the adjacent nodes to be of different colors. First, we make a list of nodes and a list of colors. Now we put the first color on the first node and move on to the adjacent node. Since the second node can't have the same color as the first, the algorithm looks for the next color on the color list and goes on till all the vertices are colored as required. When the algorithm reaches a node and can't find any color that works, the algorithm backtracks and colors the previous node with a different color. All the.

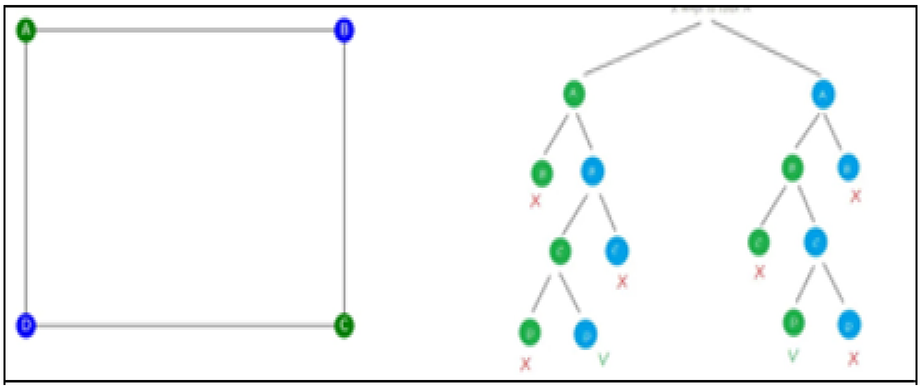

Fig 6 A Simple Graph Coloring Problem with a Square

➢ *Grover's Algorithm*

Say we have a list of unsorted data consisting of N elements and we want to find some desired elements from that list. Mathematically, we can define this problem as if we have a function f such that, f(x)=1 if x is marked (desired solution) f(x)=0 Otherwise A classical algorithm will, on average, check the list for N/2 times to find the desired element. In 1996, Lov Grover proposed Grover's Algorithm that could find the desired elements in O N evolutions (Grover, 1996). Grover's algorithm uses quantum mechanical properties: superposition and interference and provides a quadratic speedup for searching unstructured databases (Saha et al., 2015).

There are three parts of Grover's algorithm: Ini initialization, Oracle, and Diffuser. Firstly all the qubits, which are used to translate the problem we want to solve using Gorver's algorithm, are applied to the Hadamard gate which creates a superposition of all the possible states with equal amplitude. This step is called Initialization.

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{1}$$

In the second step, the oracle function is applied to these states in superposition (Ket s). The Oracle Uw is designed such that it can mark the desired item(s) in the database by flipping the

$$U_f |x\rangle = \begin{cases} -|x\rangle & \text{if } x \text{ is a solution state,} \\ \;\;|x\rangle & \text{otherwise.} \end{cases} \tag{2}$$

Geometrically, the oracle Uw applied to ket s can be visualized as a reflection around the set of orthogonal states to ket w (which is the desired state) written as,

$$U_w = I - 2|w\rangle\langle w| \tag{3}$$

An important thing to note is that the oracle is problem-based so for each different problem to be solved we need to design the oracle accordingly.

In the third step, after applying the Oracle function, we apply the diffusion operator, which reflects the amplitudes of the states about the mean of all the states. The diffusion operator does the work of amplifying the amplitude of the solution state(s) and suppressing the amplitude of the non solution states.

$$U_d = 2|s\rangle\langle s| - I \tag{4}$$

Core working of Grover's Algorithm. So following the measurement, the solution state is easily noticeable with the highest amplitude. We easily find the answer we desire.

➢ *Graph Coloring Circuit*

There are several methods of implementing Grover's search algorithm in a quantum circuit. For our use, we developed a basic form of a quantum circuit to implement Grover's Search algorithm to solve the graph coloring problem for planar graphs.

Our paper will present an implementation of Grover's Search algorithm to solve the graph coloring for a simple planar graph with 4 nodes and 4 edges (see Fig: 7). We will also present a more general method that works for any planar graph with n nodes and e edges between them.

Suppose, for simplicity's sake, we have the following graph:

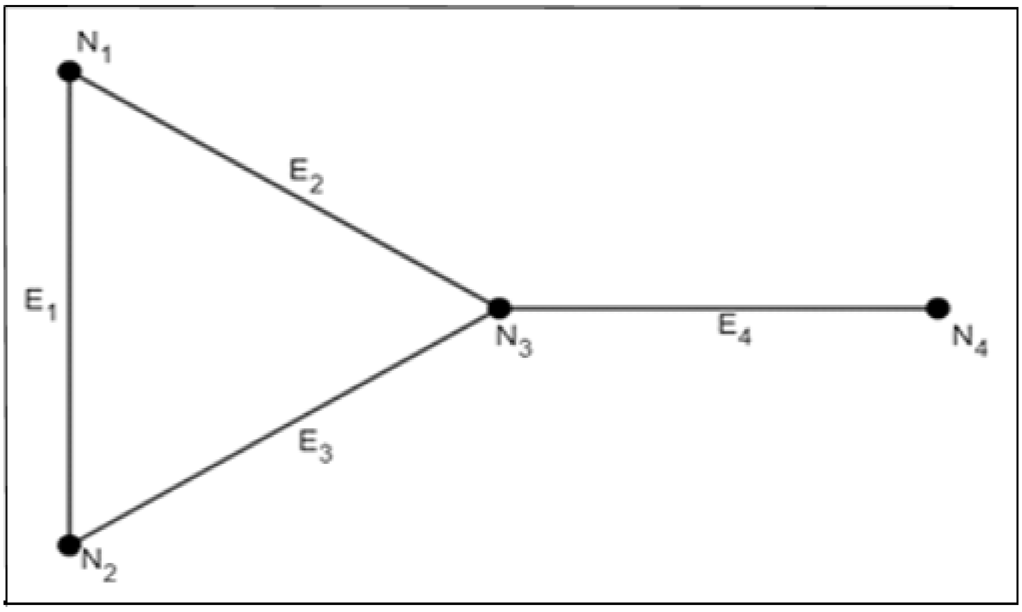

Fig 7 A Simple Graph to Illustrate Quantum Circuit. of Applying u w and v d on s . and it Turns Out

Elements in the database, and M is the number of marked states we need (Saha et al., 2015), it has the net effect that the amplitude of the desired answer is almost 1, while the amplitudes of undesired answers reduce to almost 0. With each iteration, the solution state is getting larger in am plitude, and vice versa with the other states–the It's proven that any planar graph can be colored using 4 colors (Appel & Haken, 1977). Suppose we have the following four colors with a two-bit binary number representation: Red corresponds to 00, Blue to 01, Green to 10, and Yellow to 11 binary digits. Adding binary representations to these colors, we proceed with the graph coloring circuit with the binary digits, corresponding to the colors, as input.

Our problem now is to assign these two-bit binary numbers (or simply these colors) to the $n = 4$ nodes such that no two adjacent nodes have the same binary numbers; this ensures that no two adjacent nodes have the same color. To accomplish that, let's develop a system to keep track of the different nodes, their colors, and the information between adjacent nodes.

As a first step, we can take $2n = 8$ qubits and understand that the 4 back-to-back qubit pairs represent each of the $n = 4$ nodes with the first two qubits representing N_1.

This way, the four pairs of qubits' values can be interpreted as the color in the four nodes. Suppose for now that we have these eight qubits with their values as 01000110 Because this representation has the first two qubits' value as 00, so (N1) is colored blue. Likewise, this representation implies that nodes (N2), (N3), and (N4) are colored red, blue, and green respectively. In a circuit, that would be.

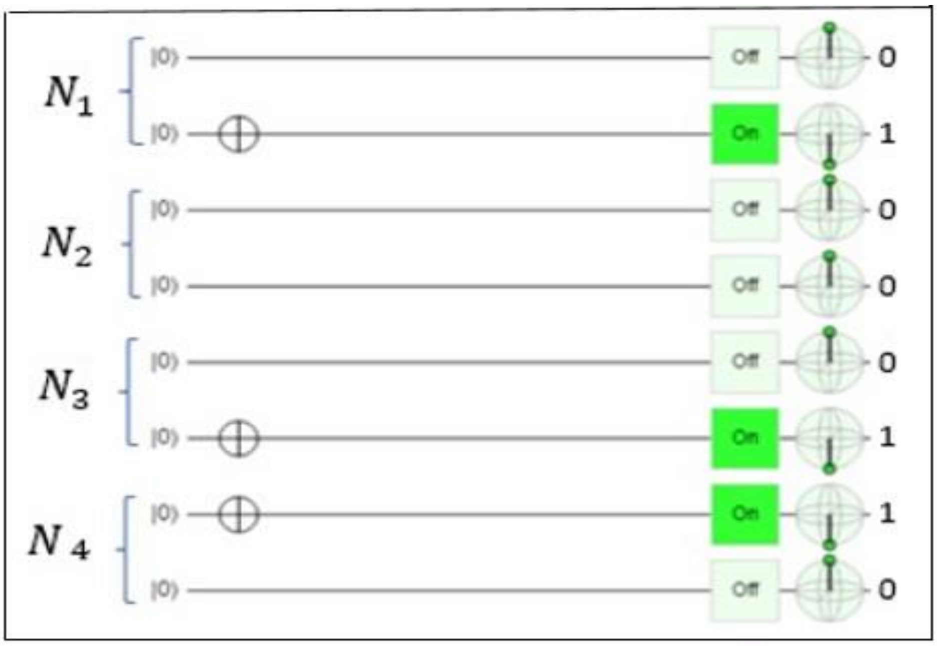

Fig8 Quantum Circuits Showing Input and Output in Binary Digits

Notice how after applying Grover's algorithm, our final output will be a $2n$-bit binary number whose leftmost two values represent N_1's color, then the next two values represent N_2's color, and so on until the last two values represent N_n's color. For our use, we'll refer to the qubits represented by the N_n node as N_n qubits. So, the second last pair of qubits is the N_3 qubits.

Since these are the only qubits that represent our color, we can apply the Hadamard H gate to make them into a uniform superposition. Any ancilla qubit we need/use later will not need to be initialized because their final values are of no use to us.

We now move on to designing the oracle to invert the phase of the coloring combination in which adjacent nodes have different colors. To achieve that, we break the oracle's function into two parts, so that the gate implementation is simpler. First, we need to identify the solution states whose adjacent nodes are differently colored. Second, we need to invert the phase of the identified solution states.

For now, let's just look at N_1 and N_2 from our specific case. Since they're adjacent, the first part of the oracle must identify if they have the same color. We'll use multi-control Toffoli gates in the following combination (see Figure 10) to see if they have the same color and store their value in an ancilla, which we'll call an edge ancilla. Note that the two nodes below are represented by two qubits each. The ancillia starts from the null state. Ancilla indexing is the same as the edge indexing from the graph. This is done for simplicity's sake.

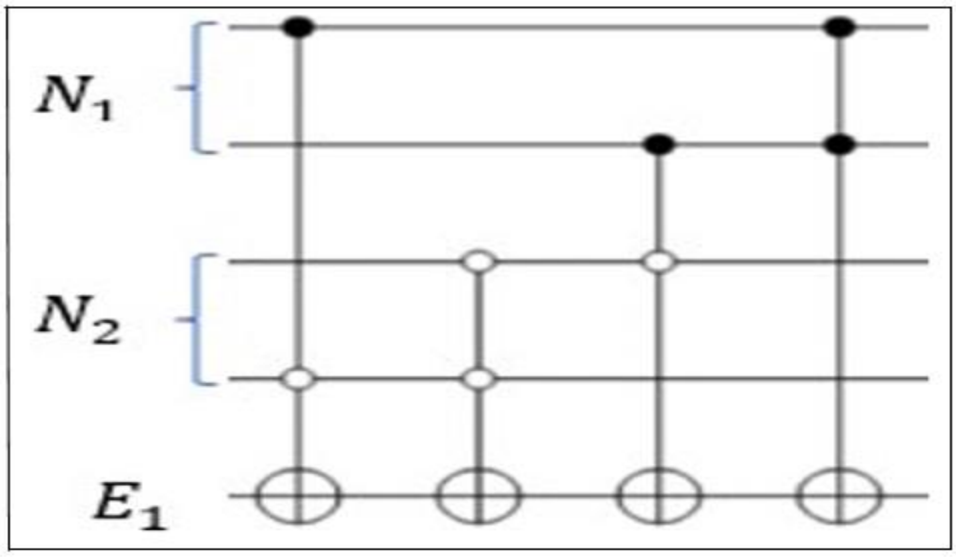

Fig 9 Circuit with two Nodes Showing Multi -Control Tofolli Gates and Ancilla

A truth table with all possible inputs in the N_1 and N_2 qubits shows that the edge ancilla returns 0 only when the N_1 and N_2 qubits have different values (i.e., different colors). This will be the case we desired. When the colors in the adjacent nodes are the same, only one of the Toffoli of the four flips the sign of the edge ancilla, making it a 1, a case that shows that the pairing isn't valid as a solution to the graph coloring problem. Else, two Toffoli gates are activated which

Act like inverses of one another to give an output of 0 in the edge ancilla.

Because of their frequent use, the combination of these Toffoli gates to check if any pair of adjacent nodes N_i and N_j with edge E_k have different colors or not, we'll be calling them color-checking gates ($C(i, j)$) and their corresponding edge ancilla is E_k. For ease of notation, we'll name the inverse of $C(i, j)$ to be $C(j, i)$ because $C(j, i)$ must have the Toffoli gates in the reverse order of $C(i, j$

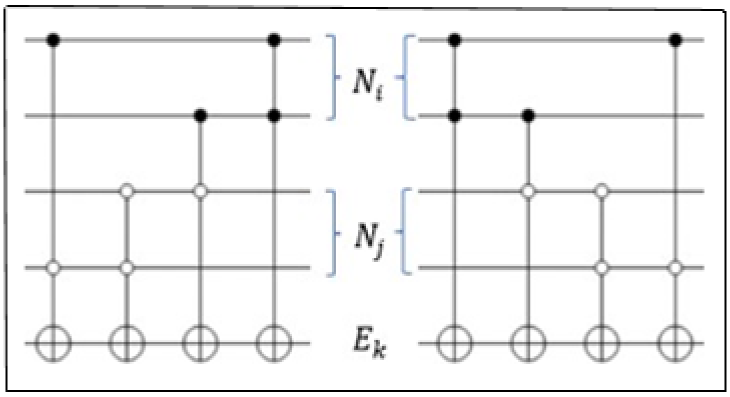

Fig 10 Toffoli Gates for Checking Colors in Pairs of Adjacent Nodes

We now design the second part of the oracle to flip the phase of the correct states identified in the first part. Our goal is to filter out the solution state after kicking its phase. For that, we'll use phase kickback (Alsing & McDonald, 2011) by using a new ancilla, which we'll refer to as the negative ancilla. In a Tofolli gate, the state of the target qubit is flipped if and only if the con trol qubit is in the state —1⟩, known as phase kickback. The kickback helps identify our solution state. We'll again use a multi-control Toffoli gate to flip the phase of the solution states.

In the first part of the oracle, we know we'll have *e* edge ancillas. If all of these edge ancillas return 0, we know that is a valid coloring because the edge ancillas are 0 only when adjacent nodes have different colors.

We now take the edge ancillas as the control and the negative ancilla as the target in a Toffoli gate to flip the phase only when all the edge ancillas return 0–when the coloring is valid. However, computationally, it is more efficient to make the controls that check 1, else we'll need to use *X* gates before and after all the controls. This will increase the gate complexity, and we try to avoid this.

Hence, in the initialization step itself, we make all

the edge ancillas to 1 to reduce the cost of the quantum circuit. In a diagram, we would have.

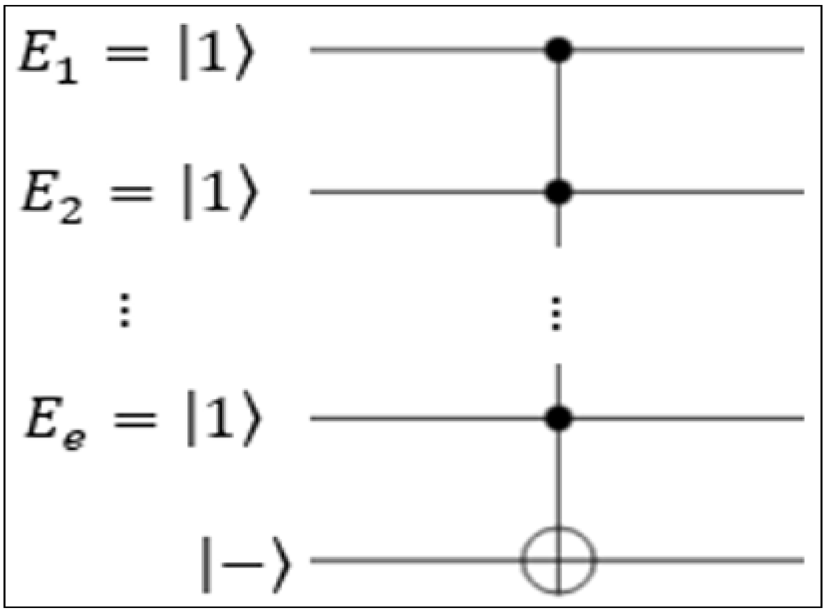

Fig 11 Initialized Edge Ancillia

So, up until now, we have initialized the states, checked which states have valid coloring, and flipped the phase of states with valid coloring. To prevent the effect on phase due to C(i,j), we apply C(j, i) right after the phase kickback completes our oracle.

- *For the diffusion operator, recall that it can be written as.*

$$s = H^{\otimes n} |0\rangle \tag{5}$$

$$D = 2\, |s\rangle \langle s| - I \tag{6}$$

$$D = 2(H^{\otimes n} |0\rangle) \langle H^{\otimes n} |0\rangle| - I \tag{7}$$

- *Because D can be written as a reflection about Mathematically, we want a gate such that.*

$$M_0 |0\rangle = |0\rangle \tag{8}$$

$$M_0 |x\rangle = -|x\rangle \quad \text{for } x \neq 0 \tag{9}$$

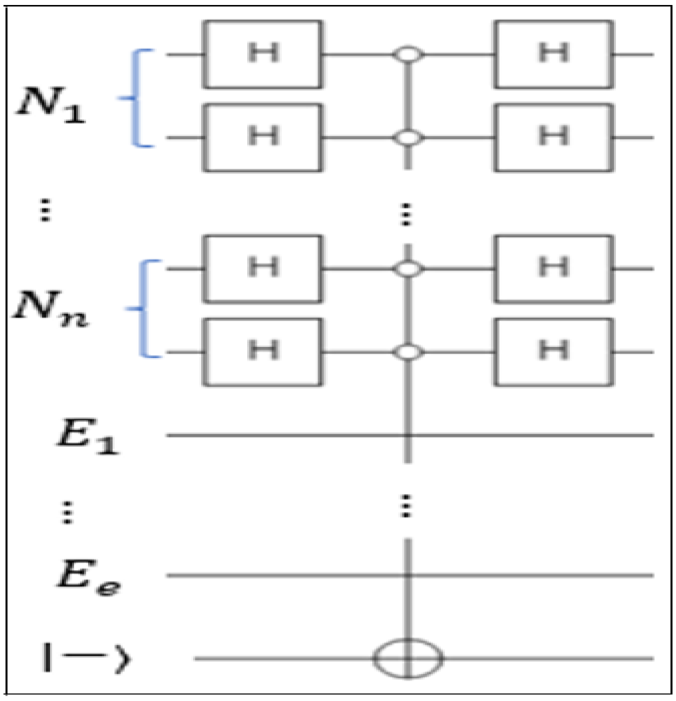

Fig 12 A Part of Initializing the Circuit

➢ *As a Pseudo Circuit, our Final Circuit would Look Like:*

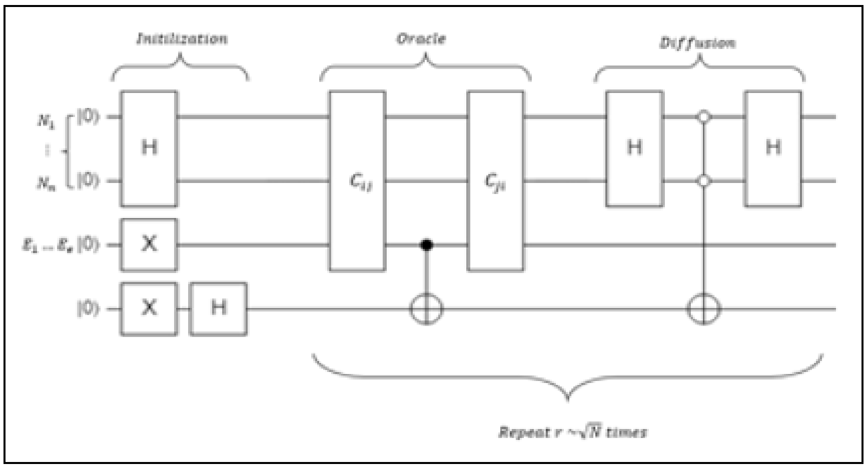

Fig 13 The Final Quantum Circuit

III. CONCLUSION

Grover's Search Algorithm has practical applications beyond sorting or searching, such as in optimizations, one of which is the NP-hard planar graph coloring problem. The implication of the graph coloring problem is vast: scheduling problems, molecular chemistry, puzzle solving, traffic signal optimization, cartography, and so on. In our case, we took the graph coloring problem of specialists in different hospitals in Nepal. While the problem has been solved using classical backtracking or greedy methods, Grover's technique offers a quadratic speedup, making it handy when the number of nodes grows larger. Our Quantum Circuit is open to further optimization to reduce the error complexity in gates and yield even more precise results.

REFERENCES

[1]. Alama, J. (2009). Euler's Polyhedron Formula. Formalized Mathematics, 16(1), 7 17. https://doi.org/10.2478/v10037 008 0002 6.
[2]. Alsing, P. M., &amp; McDonald, N. (2011). Grover's search algorithm with an entangled database state. Proc. SPIE 8057, Quantum Information and ComputationIX,80570R https://doi.org/10.1117/12.883092.
[3]. Berman, L. W., &amp; Williams, G. I. (2009). Exploring Polyhedra and Discovering Euler's Formula. Resources for Teaching Discrete Mathematics. Mathematical Association of America. Boyer, M., Brassard, G., Høyer, P., &amp; Tapp, A. (1998). Tight bounds on quantum searching. Fortschritte der Physik: Progress of Physics, 46(4 5), 493 505.
[4]. Diao, Z. (2010). Exactness of the Original Grover Search Algorithm. Physical Review A, 82. https://doi.org/10.1103/PHYSREVA.82.044301.
[5]. Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. Proceedings of the Twenty Eighth Annual ACM Symposium on Theory of Computing STOC '96. https://doi.org/10.1145/237814.237866.
[6]. K. Appel, W. Haken "Every planar map is four colorable. Part I: Discharging," Illinois Journal of Mathematics, Illinois J. Math. 21(3), 429 490, (September 1977).
[7]. Lack of superspecialist doctors threatens the future of Nepal's health sector OnineKhabar English News. (2023,February13). https://english.onlinekhabar.com/lack superspecialist doctors nepal html.
[8]. Mukherjee, S. (2022, February 8). A Grover search based algorithm for list coloring.
[9]. Saha, A., Chongder, A., Mandal, S. B., &amp; Chakrabarti, A. (2015, December). Synthesis of vertex coloring problem using Grover's algorithm. 2015 IEEE . International Symposium on Nanoelectronic and Information Systems, pp. 101 106. IEEE.
[10]. Wilf, H. S. (1984). Backtrack: An O(1) expected time algorithm for the graph coloring problem. Information Processing Letters, 18(3), 119 121. https://doi.org/10.1016/0020 0190(84)90013 9