Quantum-Assisted Graph Coloring: Solving the Graph Coloring Problem with Grover’s Algorithm in Qiskit

**Abde Manaaf Ghadiali1, Rahul Simha2**

1Graduate Student, Quantum Computing Research, George Washington University, United States  
2Professor, Department of Computer Science, George Washington University, United States

## Abstract

The graph coloring problem is a fundamental combinatorial optimization challenge with applications in scheduling, register allocation, and frequency assignment [1]. Classical algorithms often struggle with large-scale instances due to their high computational complexity [2]. This paper presents a quantum approach leveraging Grover’s search algorithm to efficiently identify all valid graph colorings. Instead of encoding all possible color assignments, we encode only the restricted color pairs that satisfy the graph's constraints, reducing the search space while maintaining quantum parallelism. We implement a Qiskit-based quantum oracle, applying amplitude amplification to filter valid solutions from a superposition of restricted states. Experimental results on small graph instances demonstrate the feasibility of this approach, showcasing a potential quantum speedup over classical brute-force methods [1]. However, scalability remains a challenge due to quantum hardware limitations and decoherence effects [2]. This work contributes to the advancement of quantum combinatorial optimization and lays the foundation for future research on quantum-enhanced graph algorithms.

## Introduction

The graph coloring problem is a fundamental combinatorial optimization problem with significant applications in various domains, including scheduling, register allocation, frequency assignment, and network optimization [3]. Given a graph the objective is to assign colors to each node in such that no two adjacent nodes share the same color, using the minimum number of colors possible. This problem is known to be NP-hard, making it computationally intractable for large instances when solved using classical brute-force or heuristic approaches [4].

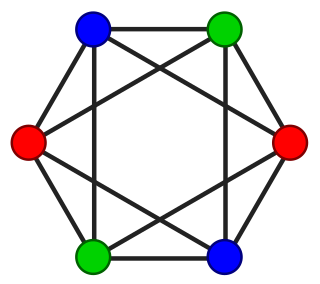


Figure 1 - A sample of Graph Coloring Problem, source - https://en.wikipedia.org/wiki/Graph\_coloring

Classical algorithms for graph coloring generally fall into two categories: exact algorithms, which guarantee an optimal solution but suffer from exponential time complexity, and heuristic or approximation algorithms, which provide faster but suboptimal solutions [5]. While techniques such as backtracking, constraint satisfaction, and greedy heuristics can handle small to moderate-sized graphs efficiently, they struggle with scalability due to the exponential growth of the solution space. This limitation motivates the exploration of quantum computing as a potential alternative for solving combinatorial optimization problems [6].

Quantum computing leverages superposition, entanglement, and quantum parallelism to process large solution spaces more efficiently than classical methods. One of the most powerful quantum algorithms for unstructured search problems is Grover’s search algorithm, which provides a quadratic speedup over classical brute-force search by amplifying the probability of correct solutions [7]. By formulating the graph coloring problem as a search problem, Grover’s algorithm can be used to efficiently identify valid color assignments that satisfy graph constraints.

This paper explores the application of Grover’s search algorithm to the graph coloring problem, leveraging quantum amplitude amplification to find all valid colorings in a computationally efficient manner [2]. We implement this approach using Qiskit, encoding only restricted color pairs rather than all possible color assignments to optimize the search process. The proposed method is evaluated on small graph instances, demonstrating the feasibility of using quantum search for combinatorial optimization. Despite hardware limitations and scalability challenges, this work contributes to the growing field of quantum combinatorial optimization, paving the way for future advancements in quantum-enhanced graph algorithms [2].

## Background and Related Work

3.1] Classical Approaches to Graph Coloring

The graph coloring problem has been extensively studied in combinatorial optimization. Traditional methods include Exact Algorithms, which guarantee optimal solutions but often exhibit exponential time complexity, making them impractical for large graphs [8]. Additionally, Heuristic and Approximation Algorithms, such as greedy algorithms and local search heuristics, provide faster but potentially suboptimal solutions. Recent surveys have explored various methods and applications, highlighting ongoing challenges in efficiently computing the chromatic number for arbitrary graphs [9].

3.2] Introduction to Grover’s Search Algorithm

Grover's algorithm, introduced in 1996, is a quantum algorithm designed for unstructured search problems [10]. It offers a quadratic speedup over classical brute-force search methods by amplifying the probability amplitude of the desired solution [11]. Specifically, for a search space of size containing valid solutions, Grover's algorithm finds a correct solution in approximately iterations. This provides a quadratic speedup over classical brute-force search, which requires steps on average [12] [Appendix A].

3.3] Previous Work on Quantum Graph Coloring

The integration of quantum computing into graph coloring has gained significant attention in recent years, leading to several notable advancements [13]. Research has demonstrated quantum algorithms providing speedups for combinatorial problems such as Set Partition, Set Cover, and Set Packing, leveraging classical enumeration algorithms that are amenable to quadratic quantum speedups, indicating potential applications to graph coloring [14]. Additionally, studies have introduced exponential-space quantum algorithms capable of computing chromatic numbers with improved running times using quantum random access memory (QRAM) [15]. These approaches employ Grover’s search within branching algorithms, highlighting the feasibility of quantum techniques for solving graph coloring problems. Furthermore, the development of hybrid quantum-classical algorithms, such as the Recursive Quantum Approximate Optimization Algorithm (RQAOA), has been applied to problems like MAX-k-CUT [16]. These hybrid methods combine quantum and classical techniques to approximate solutions for graph coloring, offering potential advantages over purely classical or quantum approaches.

3.4] Overview of Qiskit and Quantum Circuits

Qiskit is an open-source quantum computing framework developed by IBM that enables the design, simulation, and execution of quantum circuits on both simulators and actual quantum hardware [17]. It consists of several key components that facilitate quantum computing research and applications. Aer offers high-performance simulators for testing quantum circuits without the need for physical quantum processors. In the context of graph coloring, Qiskit enables researchers to implement quantum algorithms, such as Grover’s search, to explore potential speedups and efficiencies over classical methods [18].

## Problem Formulation

The graph coloring problem involves assigning colors to the vertices of a graph such that no two adjacent vertices share the same color, while minimizing the number of colors used. This minimum number of colors required is known as the chromatic number . We aim to determine all valid color assignments for the graph using exactly colors.

To translate this problem into a quantum search problem, we encode color assignments into quantum states. Consider a simple 2-node graph with one edge connecting the two nodes. Since adjacent nodes must have different colors, at least two colors are required. Suppose we use black and red, where we encode the first node as (black) and the second node as (red) using two qubits per node. The valid solutions for this problem are and (following Qiskit’s right-to-left qubit ordering), meaning these states should have the highest probability amplitudes in the quantum superposition while other invalid states should be suppressed.

The oracle function for this problem is implemented as a similarity quantum circuit (***SIM\_OP*** block), which checks if the first node’s color qubits (e.g., ​) are identical to those of the second node (​). If either pair produces a 1-state output, indicating a valid coloring, the phase of the corresponding quantum state is flipped. This phase inversion enables Grover’s amplitude amplification to enhance the probability of valid solutions.

To find the correct solutions, we apply the oracle and Grover’s diffusion operator approximately ​ times, where is the total search space and is the number of valid solutions. However, since is unknown for large graphs, we perform adaptive iterations, starting from one iteration and increasing up to iterations. Finally, quantum simulation using Qiskit AER is used to measure the states with the highest probability amplitudes.

It is important to note that restricting the superposition to only valid encoded color pairs require additional quantum circuit components, which are not currently factored into the time complexity analysis. This presents an area for further optimization and exploration in quantum graph coloring algorithms.

## Methodology: Implementing Grover’s Algorithm in Qiskit

5.1] Qubit Representation of Graph Coloring

In this methodology, we encode the graph coloring problem into a quantum representation, where each node's color is mapped to a specific quantum state. The number of qubits required per node depends on the total number of distinct colors used in the graph. For a graph requiring up to colors, we use at least qubits per node to represent all possible color assignments.

**5.1.1] Encoding Graph Nodes and Colors into Qubits**

Each color is assigned to a unique computational basis state. For example, in a 2-node, 1-edge graph, where two colors are sufficient, we can encode, for black and, for red. For larger graphs, such as a 3-node, 2-edge or 3-node, 3-edge graph, requiring three colors, we extend the encoding:

* for black
* for red
* for green

If more than four colors are needed, additional qubits are required. For example, a 5-color graph would be encoded using 3-qubit triplets to accommodate all colors within a computational basis. For consistency, we shall use two qubits for the 2-node, 1-edge graph to represent the node colors, even though only one qubit is sufficient, to maintain a uniform qubit structure across different graph sizes and simplify circuit design in later stages.

**5.1.2] Restricting the State Space for Efficient Computation**

A major challenge in implementing Grover’s search for graph coloring is ensuring that only valid color assignments enter the oracle and diffusion operators. Without restrictions, a superposition of all possible states would be inefficient and would include invalid color assignments, reducing the probability of measuring a correct solution. To overcome this, we employ state restriction techniques, including:

1. **Basis Encoding**: A straightforward method where only allowed states are initialized in the quantum register, preventing invalid color combinations from entering the computation. However, this method can be resource-intensive when implemented directly.
2. **Custom Quantum Circuit for State Filtering**: Instead of basis encoding, we construct a custom quantum circuit that filters out undesired states from the superposition. This approach ensures that only valid color combinations propagate through the oracle and diffusion operators, improving Grover’s efficiency.

In this study, we focus on small graph instances (2-node, 1-edge and 3-node, 2-edge/3-edge graphs) to illustrate our encoding scheme effectively. We implement custom quantum circuits to restrict the state space for 3-node graphs, ensuring that only valid color sets remain in superposition. Figure 2 presents the quantum circuit design for filtering valid states in a 2-node 1-edge graph. By applying these encoding techniques, we reduce the search space, improve computational efficiency, and increase the probability of measuring valid solutions in the final quantum state.

A diagram of a circuit

AI-generated content may be incorrect.

Figure - Quantum Circuit to encode colors of nodes to set of qubits, restricting the colors to only two

5.2] Oracle Construction

The oracle function plays a critical role in Grover’s search algorithm, as it marks valid graph colorings by flipping their quantum phase, enabling amplitude amplification in later steps. In this section, we design the oracle to recognize valid color assignments and implement it using Qiskit.

**5.2.1] Designing the Oracle to Mark Valid Colorings**

For a 3-node, 3-edge graph (as shown in Figure 3), we require a minimum of three colors. Suppose we encode these colors as: for black, for red, for green.

A diagram of a network

AI-generated content may be incorrect.

Figure 3 - 3-node, 3-edge Graph. This graph will require a minimum of three colors.

Since each node requires two qubits for encoding, a total of six qubits are required to represent node colors. Additionally, we introduce one ancilla qubit per edge to verify valid colorings and one output qubit to store the oracle’s result.

Thus, the total number of qubits required is:

For a 3-node, 3-edge graph, this results in: .

**5.2.2] Circuit Implementation in Qiskit**

The oracle function operates in two main stages:

*Stage 1: Applying the Similarity Operation (SIM\_OP Block)*

The SIM\_OP block is a quantum circuit that determines whether two adjacent nodes share the same color. This is achieved using CNOT (CX) gates and an OR Block, as follows:

* Comparing Corresponding Qubits of Adjacent Nodes: For each edge , we apply a CX gate controlled by node ’s first qubit and targeted at node ’s first qubit. Another CX gate is applied to compare node ’s second qubit with node ’s second qubit. Since a CX gate behaves like a classical XOR operation, the result is 0 if the qubits are the same and 1 if they are different.
* Using an OR Block to Determine Validity: The outputs of the CX gates are passed through an OR Block (a quantum subcircuit). The OR Block ensures that if either of the two CX results is 1, it outputs 1, indicating that the nodes have different colors. If both results are 0, the OR Block outputs 0, meaning the nodes share the same color (an invalid solution). The additional edge qubits are used as ancilla qubits within the OR Block to store these intermediate results. Figure 4 illustrates the implementation of the SIM\_OP block and OR Block quantum circuits.

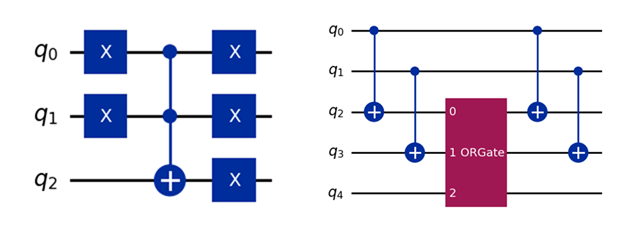


Figure 4 - The OR Block Quantum Circuit (Left) and the SIM\_OP Block (Right) using the OR Block as a gate.

* Applying Similarity Checks to All Edges: We repeat this process for all edges, storing the results in their corresponding edge qubits.

*Stage 2: Phase Flip for Valid Solutions*

Once the OR Blocks determine whether all edges satisfy the coloring constraints, we apply a multi-controlled phase flip to mark valid states.

* Applying the Multi-Controlled X (MCX) Gate: The MCX gate (multi-controlled NOT gate) is used to flip the phase of any quantum state where all OR Block outputs are 1 (indicating a valid coloring). The output qubit serves as the target qubit for the MCX gate.
* Surrounding the MCX Gate with Hadamard Gates: We apply Hadamard (H) gates before and after the MCX operation on the target qubit to ensure a phase inversion for the marked state.

Thus, if our oracle function is applied to an initial state (valid coloring state):

Then, the oracle flips the phase:

**5.2.3] Un-computation and Resetting of OR Blocks**

To maintain coherence in the quantum circuit, we must reverse intermediate computations after applying the phase flip. This involves reapplying CX gates on node qubits to revert their state to the original superposition and resetting OR Block results to free the ancilla qubits for future iterations of Grover’s search. This ensures that each iteration of Grover’s algorithm starts with a clean computational state, improving the efficiency of amplitude amplification.

Diffusion Operator

The **diffusion operator**, also known as the **inversion-about-the-mean operator**, is a crucial component of **Grover’s search algorithm**. Its primary role is to **amplify the probability amplitudes** of the states marked by the oracle (i.e., valid colorings in our case) and suppress the amplitudes of invalid solutions. This is achieved by reflecting the current quantum state across the average amplitude of the superposition, thereby increasing the likelihood of measuring a correct solution after repeated Grover iterations.

**Mathematical Formulation of the Diffusion Operator**

The diffusion operator DDD is implemented as a unitary transformation defined by the following circuit structure:

D=eiπ⋅H⊗n⋅X⊗n⋅H⋅MCX⋅H⋅X⊗n⋅H⊗nD = e^{i\pi} \cdot H^{\otimes n} \cdot X^{\otimes n} \cdot H \cdot \text{MCX} \cdot H \cdot X^{\otimes n} \cdot H^{\otimes n}D=eiπ⋅H⊗n⋅X⊗n⋅H⋅MCX⋅H⋅X⊗n⋅H⊗n

Where:

* H⊗nH^{\otimes n}H⊗n: Hadamard gates applied to all computational basis qubits (excluding ancilla).
* X⊗nX^{\otimes n}X⊗n: Pauli-X (NOT) gates applied to flip the basis states.
* MCX\text{MCX}MCX: A **multi-controlled X (Toffoli-like) gate** that flips the state of the output qubit when all control qubits are in the ∣1⟩|1\rangle∣1⟩ state.
* eiπe^{i\pi}eiπ: A **global phase** equivalent to multiplying the entire transformation by −1-1−1, aligning with the standard Grover formulation.

**Circuit Implementation in Qiskit**

To build the diffusion operator in **Qiskit**, we follow these steps:

1. **Apply Hadamard Gates**  
   Apply **Hadamard gates** to all relevant qubits involved in the encoding of node colors. Note that **ancilla qubits** used in the oracle (e.g., for the OR blocks or intermediate results) are excluded from this operation.
2. **Apply Pauli-X Gates**  
   Apply **X gates** to the same qubits (excluding ancillas), inverting the basis states to prepare for the MCX reflection.
3. **Apply Hadamard and MCX**
   * Apply a **Hadamard gate** to the **output qubit**, which was used in the oracle to mark valid solutions.
   * Use all the **computational qubits** (excluding the output and ancillas) as **controls** for the **MCX gate**, with the **output qubit as the target**.
4. **Reverse the Transformations (Uncomputation)**  
   After the MCX gate, we **uncompute** the earlier operations by reapplying them in reverse:
   * Apply **Hadamard** to the output qubit.
   * Apply **X gates** to the control qubits.
   * Apply **Hadamard gates** again to the same qubits.
5. **Apply a Global Phase**  
   For completeness and to align with Grover’s original formulation, we apply a **global phase of eiπ=−1e^{i\pi} = -1eiπ=−1**, ensuring that the diffusion operator reflects amplitudes about the mean with the correct sign.

The **diffusion operator must act only on the superposition space** representing encoded node colors. Including ancilla or output qubits can distort the operation, so care must be taken to isolate the relevant subset. The **MCX gate** is central to amplitude amplification. In practice, Qiskit uses **ancilla-assisted decomposition** of the MCX gate, or supports ancilla-free methods depending on the circuit depth and qubit availability. Applying this operator multiple times (in alternating sequence with the oracle) leads to **constructive interference** of valid solutions, enabling Grover’s quadratic speedup.

In the **next section**, we will demonstrate a complete example of applying both the oracle and diffusion operator in a working Qiskit implementation.

Quantum Circuit Design

Now we have our oracle and diffusion operator ready for implementing the Grover’s search algorithm. We know that we need to apply the grover search algorithm for approximately iterations. Lets say we are working on a 2 node 1 edge graph. Hence, we would need:

Which is qubits. We also know that for coloring this graph, we need a maximum of 3 colors. Lets denote this as |00> (black), |01> (red) and |10> for green. Then we apply or custom encoding circuit to get the superposition of all the states with the restricted colors. Lets call this operator E. Then:

Psi\_1 = |000000000>

Psi\_2 = E|000000000> =

 4.4 Quantum Circuit Design

* Full Qiskit Implementation of Grover’s Search

 4.5 Measurement and Interpretation of Results

* Extracting Solutions from Quantum Simulations

**Experimental Results and Analysis**

 Test Cases and Simulated Graphs

 Performance Comparison with Classical Approaches

 Analysis of Success Probability and Required Iterations

 Scalability and Hardware Constraints

**Discussion and Limitations**

 Strengths and Benefits of Quantum Graph Coloring

 Limitations of Grover’s Approach for Large Graphs

 Noisy Quantum Hardware Considerations

 Possible Improvements and Hybrid Approaches

**Conclusion and Future Work**

 Summary of Findings

 Potential for Real-World Applications

 Future Research Directions (e.g., Variational Quantum Approaches, Hardware Optimizations)

## Appendix

A] Grover’s Search Algorithm

Grover’s algorithm is a quantum search algorithm that provides a quadratic speedup for searching an unsorted database of elements, requiring approximately queries instead of in classical search. The algorithm amplifies the probability amplitude of the correct solution(s) through iterative applications of an oracle function and a diffusion operator, allowing us to find a marked state in significantly fewer steps than classical methods. Grover’s algorithm consists of three main steps:

**Initializing the Equal Superposition State**

The algorithm begins by initializing an -qubit system in the state and then applying a Walsh-Hadamard transformation to create an equal superposition of all possible states in the search space. This is represented as:

Applying the Hadamard gate to each qubit transforms the system into an equal superposition:

Where represents the decimal equivalent of each possible binary state. This initialization ensures that each state has an equal probability amplitude before applying Grover’s search.

**Applying the Oracle to Mark the Solution**

The oracle function is a quantum operation that marks the correct solution by flipping its phase. The oracle is problem-dependent and encodes the solution by mapping:

Where if is the correct solution, and otherwise. This means that the oracle multiplies the amplitude of the marked state(s) by , flipping its phase by radians while leaving all other states unchanged. For example, if the correct solution is , applying the oracle to results in:

This step makes the correct solution distinguishable from the rest, preparing it for amplitude amplification.

**Applying the Grover Diffusion Operator for Amplitude Amplification**

After marking the correct state, we apply the Grover diffusion operator , which increases the probability amplitude of the marked solution. This operator effectively reflects the quantum state across the mean amplitude, enhancing the likelihood of measuring the correct answer. The diffusion operator is defined as:

Or more simply,

A diagram of a diagram

AI-generated content may be incorrect.

Figure 3 - Grover's Diffusion Operator in Qiskit

Applying modifies the amplitudes such that the probability of measuring the correct solution increases with each iteration. The complete transformation after one Grover iteration is:

**Mathematical Explanation of Amplitude Modification**

Let’s assume that the quantum state after applying the oracle is:

where:

* is the total number of states.
* is the amplitude of each state .
* For the correct state , the amplitude has been flipped by the oracle, i.e., ​​.

The mean amplitude of all states before applying is:

The Grover diffusion operator performs a reflection about this mean amplitude, modifying each amplitude ​ as follows:

This transformation increases the amplitude of the marked states while reducing the amplitude of unmarked states.

**Number of Iterations and Convergence**

To maximize the probability of measuring the correct solution, we repeat the oracle and diffusion steps approximately:

iterations, where:

* is the total search space size,
* is the number of correct solutions.

Since is typically unknown for large problems, an adaptive search approach is often used, where the algorithm starts with one iteration and gradually increases up to .

## References

[1] Shimizu, K., & Mori, R. (2019). Exponential-time quantum algorithms for graph coloring problems. *arXiv preprint arXiv:1907.00529*.

[2] Ferber, A., Hardiman, L., & Chen, X. (2025). Improved sublinear algorithms for classical and quantum graph coloring. *arXiv preprint arXiv:2502.06024*.

[3] GeeksforGeeks. (2024). *Introduction to Graph Coloring*. Retrieved from <https://www.geeksforgeeks.org/graph-coloring-applications/>

[4] Kumar, V., & Kumar, A. (2024). *Graph Coloring Problems and Their Applications in Scheduling*. Retrieved from <https://www.researchgate.net/publication/2910750_Graph_Coloring_Problems_and_Their_Applications_in_Scheduling>

[5] Shoaib, M. (2023). *How to Solve the Graph Coloring Problems Using Qiskit's Grover Algorithm*. Retrieved from <https://medium.com/@shoaib6174/how-to-solve-the-graph-coloring-problems-using-qiskits-grover-algorithm-e87eb52f203>

[6] Li, M. (2020). *Grover's Search Algorithm Using IBM's Qiskit*. Retrieved from <https://maggiezli.medium.com/grovers-search-algorithm-3233592820d3>​

[7] Sanyal, A., Saha, A., Saha, D., Saha, B., & Chakrabarti, A. (2020). *Circuit Design for Clique Problem and Its Implementation on Quantum Computer*. Retrieved from <https://arxiv.org/abs/2004.10596>

[8] Kubale, M. (2004). *Graph colorings*. American Mathematical Society.

[9] Jensen, T. R., & Toft, B. (2011). *Graph coloring problems*. Wiley.

[10] Karpinski, M., & Zelikovsky, A. (1997). Approximating the chromatic number of low-degree graphs. *Journal of Algorithms, 23*(2), 265-280. https://doi.org/10.1006/jagm.1997.0893

[11] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 212-219. https://doi.org/10.1145/237814.237866

[12] Shimizu, K., & Mori, R. (2022). Exponential-time quantum algorithms for graph coloring problems. *Algorithmica, 84*(3), 859-880. https://doi.org/10.1007/s00453-022-00976-2

[13] Ambainis, A., & Kokainis, M. (2022). Quantum algorithms for graph coloring and other partitioning problems. *Proceedings of the 49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, 69(1), 69:1-69:14. https://doi.org/10.4230/LIPIcs.ICALP.2022.69

[14] Bravyi, S., & Gosset, D. (2017). Improved classical algorithms for quantum many-body ground states. *Nature Communications, 8*, 15258. https://doi.org/10.1038/ncomms15258

[15] Abraham, H., AduOffei, D., Agarwal, R., et al. (2019). *Qiskit: An open-source framework for quantum computing*. Zenodo. https://doi.org/10.5281/zenodo.2562111

[16] Nannicini, G. (2019). Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Physical Review E, 99*(1), 013304. https://doi.org/10.1103/PhysRevE.99.013304

[17] Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*. <https://arxiv.org/abs/1411.4028>

[18] Ambainis, A. (2019). Quantum walk algorithm for element distinctness. *SIAM Journal on Computing, 48*(1), 1-16. https://doi.org/10.1137/15M1048233