Journal of the Korean Institute of Industrial Engineers
Vol. 51, No. 6, pp. 454-464, December 2025.
ISSN 1225-0988 | EISSN 2234-6457
https://doi.org/10.7232/JKIIE.2025.51.6.454

Grover Algorithm-based Quantum Scheduling Framework for Constrained Graph Coloring

Lutfiana Sausan · Hyunsoo Lee†

Department of Industrial Engineering, Kumoh National Institute of Technology

제약 조건이 있는 그래프 컬러링 문제를 위한 그로버 알고리즘 기반 퀀텀 스케줄링 프레임웍

Lutfiana Sausan · **이현수**

국립금오공과대학교 산업공학과

Quantum computing holds significant promise for solving NP-hard combinatorial problems by leveraging quantum properties such as superposition, entanglement, and interference. In this study, we apply Grover's algorithm to a constrained scheduling problem modeled as a graph coloring task with an added complexity: ensuring that the total processing time of all jobs assigned to a machine does not exceed its capacity. This additional constraint increases the problem's difficulty beyond traditional graph coloring. Our proposed quantum framework demonstrates the ability to generate valid schedules that satisfy both coloring and capacity constraint, highlighting the practical potential of quantum computing for addressing complex scheduling challenges.

Keywords: Quantum Computing, Optimization, Grover's Algorithm, Constrained Graph Coloring

1. Introduction

Quantum computing offers a powerful approach to combinatorial optimization (Harwood *et al.*, 2021), enabling speedups for some NP-hard problems through superposition, entanglement, and interference (Ajagekar *et al.*, 2019). Quantum computing techniques share the advantages of quantum mechanics (Oh *et al.*, 2024; Khairunissa *et al.*, 2022; Oh *et al.*, 2022). In quantum computing methodologies, Qubits can exist in superposition, allowing quantum algorithms to process multiple possibilities simultaneously (Clarke *et al.*, 2008; Lee *et al.*, 2023) and solve complex optimization problems more efficiently (Coccia *et al.*, 2024; Hong *et al.*, 2024).

One notable application of quantum computing in combinatorial optimization is the graph coloring problem, which arises in various fields such as flight gate assignment (Stollenwerk *et al.*, 2020), frequency and register allocation (Oh *et al.*, 2019), and routing (Do *et al.*, 2020). This problem involves assigning colors to the vertices of a graph such that no two adjacent vertices share the same color, with the goal of minimizing the total number of colors used.

Quantum algorithms, such as Grover's algorithm, solve graph coloring problem by exploring potential solutions in parallel (Khanal *et al.*, 2023). Grover's algorithm can be adapted for binary and ternary quantum systems (Saha *et al.*, 2015). This adaptation is crucial as it allows for a more efficient search for valid colorings in graphs, showcasing the potential for quantum speedups in combinatorial optimization tasks.

This research was supported by Kumoh National Institute of Technology(2024-2026).
† 연락저자 : 이현수 교수, 경북 구미시 대학로 61 국립금오공과대학교 산업공학과, Tel : 054-478-7661, Fax : 054-478-7679,
E-mail : hsl@kumoh.ac.kr
2025년 7월 3일 접수; 2025년 8월 13일; 2025년 9월 16일 수정본 접수; 2025년 9월 17일 게재 확정.

In this study, we extend Grover's algorithm to a scheduling problem modeled as a graph coloring task with added time constraints. This scenario reflects real-world manufacturing settings where machines have limited working hours, and tasks must be distributed to avoid overload. The challenge lies not only in assigning different machines (colors) to adjacent tasks but also in ensuring that the total processing time on each machine does not exceed its capacity.

We apply Grover's algorithm, known for its quadratic speedup, to efficiently search for valid schedules. Our oracle checks whether a candidate solution satisfies both the coloring and time constraints, ensuring adjacent tasks are on different machines and machine time limits are respected. Valid solutions are amplified through quantum amplitude amplification, enabling effective identification of feasible scheduling arrangements.

2. Background Theory

2.1 QAOA

The Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm designed to tackle optimization problems by combining classical and quantum processing (Zhou *et al.*, 2020). To solve graph coloring problem using QAOA, it is necessary to formulate the cost Hamiltonian, which represents the number of conflicts, in this case is adjacent vertices with the same color. The cost Hamiltonian is defined as:

$$H_c = \sum_{(i,j)\in E} \delta(c_i, c_j) \tag{1}$$

where $\delta(c_i, c_j)$ is an indicator function that takes the value 1 if the colors assigned to vertices i and j are the same, and 0 otherwise. This Hamiltonian captures the essence of the graph coloring problem by penalizing configurations in which adjacent vertices are assigned the same color (Lucas, 2014).

In QAOA, the quantum system is initialized in a superposition of all possible color assignments. The algorithm alternates between applying the Cost Hamiltonian H_c and a Mixer Hamiltonian H_M, which introduces transitions between different states. The mixer Hamiltonian is typically chosen to be:

$$H_M = \sum_{v\in V} X_v \tag{2}$$

where X_v is the Pauli-X operator acting on the qubit corresponding to vertex v. This operator ensures that the system explores different color assignments by flipping the qubit states.

The QAOA is parameterized by angles γ and β, which are optimized to minimize the expected value of the cost Hamiltonian (Farhi *et al.*, 2014). The quantum state after p steps of the algorithm is given by:

$$| \psi_p(\gamma,\beta) \rangle = U_M(\beta_p)\, U_C(\gamma_p \cdots U_M(\beta_1)\, U_C(\gamma_1) \,|\, \psi_0 \rangle \tag{3}$$

where $U_C(\gamma) = e^{-i\gamma H_C}$ and $U_M(\beta) = e^{-i\beta H_M}$. This process encodes the graph coloring problem into a quantum algorithm, alternating between cost and mixer Hamiltonians to gradually converge towards a solution.

Lastly, the optimization process involves finding the value of γ and β that minimize the expectation value of the cost Hamiltonian, which is expressed as:

$$\langle \psi_p(\gamma,\beta) \,|\, H_C \,|\, \psi_p(\gamma,\beta) \rangle \tag{4}$$

This expected value represents the total number of conflicts in the current solution, and minimizing it leads to an optimal or near-optimal graph coloring solution.

2.2 Quantum Annealing

In Quantum Annealing, the system starts in a simple quantum state, and then slowly evolves to the final Hamiltonian that encodes the solution to the problem. The evolution is governed by a time-dependent Hamiltonian $H_{(t)}$, which interpolates between an initial Hamiltonian H_i and the cost Hamiltonian H_c:

$$H(t) = \left(1 - \frac{t}{T}\right) H_i + \frac{t}{T} H_C \tag{5}$$

where t is the time and T is the total annealing time. The system is initialized in the ground state of H_i, which is easy to prepare. Over time, $H_{(t)}$ transitions from the initial Hamiltonian to the cost Hamiltonian H_c, with the goal of reaching the ground state of H_c, which corresponds to the optimal graph coloring configuration.

The initial Hamiltonian H_i typically takes the form of a transverse field, which promotes quantum tunneling between different states:

$$H_i = -\sum_{v\in V} \sigma_v^x \tag{6}$$

where σ_v^x is the Pauli-X operator applied to vertex v, enabling transitions between different color states for each vertex during the annealing process.

The annealing process is designed to find the minimum of the final Hamiltonian H_c, which represents the optimal or near-optimal solution to the graph coloring problem. By slowly evolving the Hamiltonian and maintaining the system in the ground state, quantum annealing exploits quantum tunneling to overcome local minima and explore the solution space efficiently.

2.3 Grover's Algorithm

Grover's Algorithm is designed for searching an unstructured database where the correct solution is known but needs to be found among multiple possibilities, such as identifying the correct coloring in a graph coloring problem (Mukherjee, 2022). Grover's algorithm leverages quantum amplitude amplification to efficiently search through the solution space, offering a quadratic speedup over classical search methods (Chakrabarty *et al.*, 2017).

The solution space for a graph with n vertices and k possible color per vertex contains k^n possible color assignments. Let the set of possible colorings be denoted as:

$$S = x_1, x_2, \cdots, x_{k^n} \tag{7}$$

where each x_1 represents a unique coloring assignment for the vertices. Grover's algorithm seeks to find the valid coloring from this set.

The essential part of Grover's algorithm is the oracle function, which identifies valid colorings by checking if adjacent vertices share the same color. The oracle applies a phase flip to the valid solutions, denoted as:

$$O(x_i) = \begin{cases} -1 \\ 1 \end{cases} \tag{8}$$

This oracle marks valid graph colorings by flipping their phase, differentiating them from invalid solutions.

Grover's algorithm begins by initializing a quantum state in a superposition of all possible colorings:

$$| \psi_0 \rangle = \frac{1}{\sqrt{k^n}} \sum_{i=1}^{k^n} | x_i \rangle \tag{9}$$

This superposition equally represents all possible color assignments. The algorithm iteratively applies two key operations: the oracle O, which identifies valid colorings, and the diffusion operator D, which amplifies the amplitude of valid solutions (Perkowski, 2022). The diffusion operator can be expressed as:

$$D = 2 | \psi_0 \rangle \langle \psi_0 | - I \tag{10}$$

where I is the identity matrix, and the operator reflects the quantum state about the average amplitude of all states.

3. Methodology

The proposed quantum scheduling algorithm is structured into a series of distinct yet interconnected stages, each contributing to the overall process of efficiently assigning tasks to machines under specific constraints. Initially, the problem is precisely defined, setting the foundation for the subsequent steps. The algorithm begins with the initialization phase, where each product is encoded in a quantum state representation. This is followed by multiple oracle constructions, which serve critical roles such as validating feasible colorings, ensuring compliance with machine time limitations, and aggregating processing times. The diffusion step then amplifies the probability amplitudes of the desired states, preparing the system for measurement. Finally, the measurement stage collapses the quantum state into a classical binary outcome, revealing the scheduling solution. <Figure 1> provides a clear overview of the sequential stages in the proposed quantum scheduling algorithm.

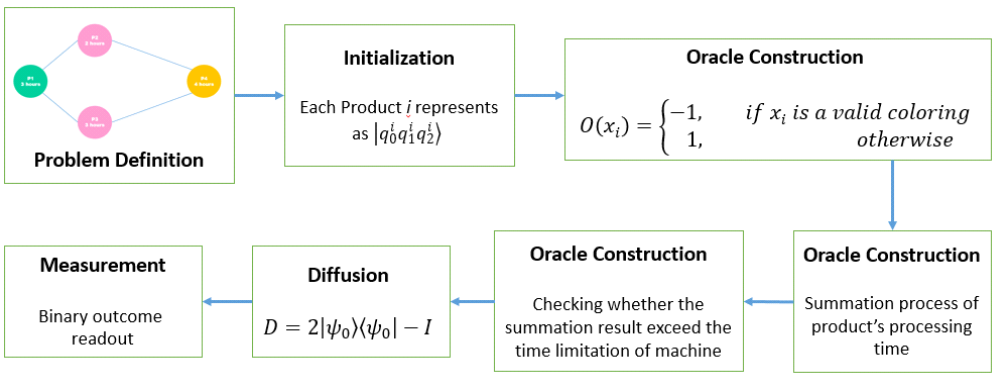


Figure 1. Flowchart of the Proposed Grover-based Scheduling Algorithm.

3.1 Initialization

In the initialization phase of Grover's algorithm, we aim to define the structure of the quantum states representing the solution space. The representation of each product is encoded using three qubits, where the first two qubits, q_0 and q_1, denote the machine assigned to the product, and the third qubit, q_2 represents the day of production.

We initialize the quantum state to represent all possible assignments of machines and production days for each product. This is achieved by preparing a uniform superposition over all possible qubit states using the Hadamard gate. For each product i, the state $|q_0^i, q_1^i, q_2^i\rangle$ represents a binary encoding of the machine and day assignment, where $q_0^i q_1^i = \{00,01,10,11\}$ corresponds to the four machines available, and $q_2^i = \{0,1\}$ encodes two possible days for production. The superposition is generated as follows:

$$|\psi_0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=1}^{2^n} |x\rangle \tag{11}$$

where $n = 3k$ (with k being the number of products), ensuring that each combination of machine and day for every product is equally probable. Thus, x takes on all possible binary combinations of machine and day assignments for all k products. the sum runs over all 2^n possible states in the superposition, representing all potential machine and day allocations for each product. In our case, $k = 4$ products, so x would take values from 000000000000 to 111111111111, representing different combinations of machine and day assignments for the products.

This initialization sets up the quantum state with all possible solutions, from which Grover's algorithm will search for the valid coloring of the graph where adjacent nodes (representing products) are assigned distinct machine-day pairs. This ensures that no two adjacent nodes share the same machine or day, as dictated by the scheduling constraints.

3.2 Oracle

In addition to ensure that adjacent nodes in the graph are assigned different colors (representing machines), our oracle incorporates a summation process to verify whether the total processing time assigned to any machine exceeds its allowable daily limit of 6 hours. This is achieved using a quantum full-adder circuit that performs binary addition to sum the processing times of products assigned to adjacent nodes.

The binary addition process is carried out by encoding the processing times as binary numbers in quantum registers. For instance, we add two products with processing times 4 hours (binary 100) and 3 hours (binary 011), resulting in a total of 7 (binary 0111). The summation is performed bit by bit using a full-adder structure. For each bit position i, the quantum circuit calculates both the sum S_i and the carry C_i for the next bit. The sum for the i-th qubit is given by the equation:

$$S_i = a_i \oplus b_i \oplus C_{i-1} \tag{12}$$

where a_i and b_i are the i-th bits of the processing times a and b, respectively, and C_{i-1} is the carry from previous bit. For the least significant bit (LSB), the carry is initialized to zero ($C_{-1} = 0$). The quantum circuit uses controlled-NOT (CX) gates to implement the XOR operation, and the sum is stored in a separate quantum register.

The carry for each bit is computed using a combination of AND and OR operations, which are implemented in the quantum using Toffoli (CCX) gates. Mathematically, the carry for bit i is expressed as:

$$C_i = (a_i \wedge b_i) \wedge (C_{i-1} \wedge (a_i \oplus b_i)) \tag{13}$$

This equation ensures that a carry is propagated if both a_i and b_i are 1, or if there was a carry from the previous bit that combines with either a_i or b_i. In the circuit, the first part $(a_i \wedge b_i)$ is implemented using a CCX gate, and the second part $C_{i-1} \wedge (a_i \oplus b_i)$ is handled by a series of CX and CCX gates.

Based on equation (12), for the LSB, the sum is calculated using the operation $S_0 = a_0 \oplus b_0$ and the carry is computed as $C_0 = (a_0 \wedge b_0)$. The circuit used for the summation process is shown in <Figure 2>. In the circuit, this is done by applying CX gates to qubits a_0 and b_0 to compute the XOR, and a CCX gate to compute the carry. For the second bit, the sum is calculated as $S_1 = a_1 \oplus b_1 \oplus C_0$ and the carry is $C_1 = (a_1 \wedge b_1) \wedge (C_0 \wedge (a_1 \oplus b_1))$. The CX gates are again used for XOR operations, while CCX gates manage the carry propagation.

The process continues for the most significant bit (MSB), where the final carry C_2 is propagated to detect any overflow. For instance, if the circuit is tasked with adding $a_2 a_1 a_0 = 100_2$ (4 hours) and $b = 011_2$ (3 hours), the result is $S_3 S_2 S_1 S_0 = 0111_2$ which equals 7 hours in decimal form. Once the summation is complete, the result is stored in a quantum register and then the quantum circuit compares the summation result to the time limitation of the machine, which is set to 6 hours.

The quantum circuit for comparison implements custom gates and controlled operations to determine whether the input number

is less than or equal to 6, marking the state as valid if the condition is satisfied. The details of the circuit are presented in <Figure 3>.

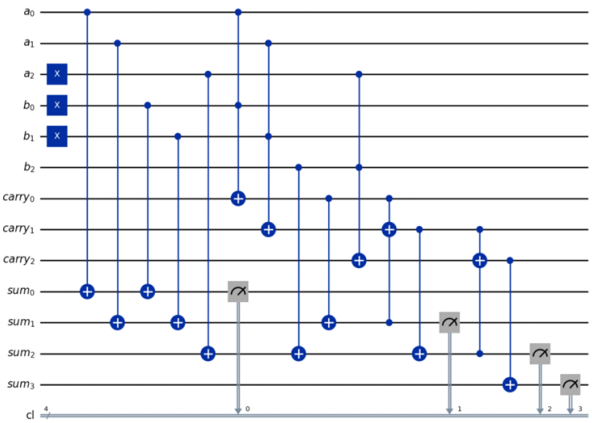


Figure 2. Quantum circuit for summation process of product's processing time.

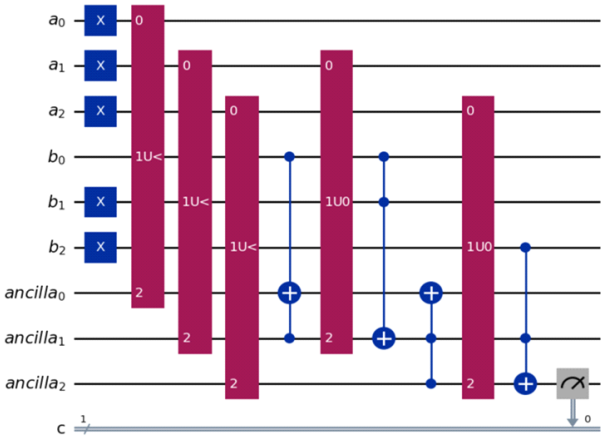


Figure 3. Quantum circuit for checking whether the summation result exceed the time limitation of each machine.

The circuit operates on two main sets of qubits, the qubits $a_2a_1a_0$ which represents the result of the previous summation operation that we aim to compare, and the qubits $b_2b_1b_0$ which are fixed to represent the binary value of 6. The circuit also involves ancilla qubits which serve as intermediate storage to facilitate the comparison process. These ancilla qubits play a crucial role in propagating information about the intermediate steps of the binary comparison. Additionally, a control qubit c is included to store the final result of the comparison and will be measured to determine if the number is valid based on the comparison criteria.

The first stage of the circuit involves preparing the binary value of 6 using X-gates. An X-gate is a quantum operation that flips the state of a qubit. Initially, the qubits are prepared in the state $|0\rangle$ and the X-gate flips the required qubits to $|1\rangle$ to match the binary representation of 6. No X-gates are applied to b_2 and b_1 since they are already in the state $|1\rangle$ but an X-gate is applied to b_0 to ensure that it is in the state $|0\rangle$ completing the binary setup for 6. The primary comparison between the two binary numbers is performed using custom quantum gates, labeled U< and U0. The details of U< gates and U0 gates can be seen in <Figure 4>.

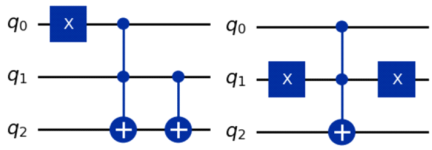


Figure 4. The gate U< (left) checks whether the binary number $a_2a_1a_0$ is less than the corresponding bits of $b_2b_1b_0$, while the gate U0 (right) checks whether the binary number is equal to 6.

The final result of the comparison is stored in the ancilla qubits. If they contain the state $|1\rangle$ this indicates that the binary number represented by $a_2a_1a_0$ is greater than 6, marking it as an invalid solution. If they contain $|0\rangle$ it means the number is less than or equal to 6.

3.3 Diffusion

The last step of Grover's algorithm is known as diffusion or amplitude amplification. This process is applied to the variable registers after the oracle has marked the correct solution by inverting its amplitude. The purpose of diffusion is to increase the probability of measuring the marked state by amplifying its amplitude and reducing the amplitudes of the non-solution states.

The diffusion operator is calculated by using (10). After each application of the oracle, the diffusion process increases the likelihood of finding the solution that does not violate the time limitation of 6 hours per machine, ensuring efficient machine scheduling.

3.4 Algorithm Overview

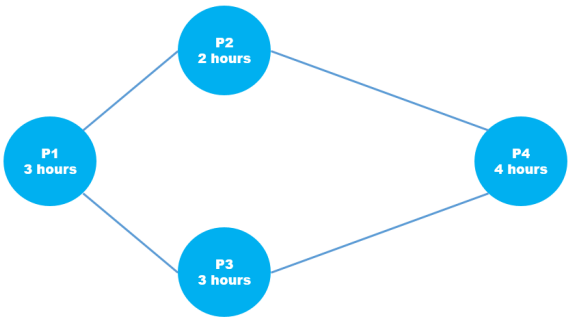
To provide a clear and concise overview of the proposed Grover-based quantum scheduling algorithm, we present its detailed procedural steps in Algorithm 1. This algorithm integrates the core components described above, including initialization, oracle construction with coloring and capacity constraints, and the diffusion operator, into a unified framework. The pseudocode

highlights the iterative nature of Grover's search adapted for the scheduling problem and illustrates how constraint checks and amplitude amplification are systematically performed to efficiently identify feasible and optimal scheduling solutions.

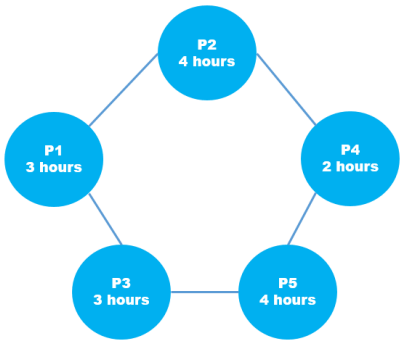
```
Algorithm 1.
Input:
  G(V, E) // Conflict graph: V = products, E = edges (conflicts)
  M // Number of machines
  D // Number of production days
  T_max // Maximum machine capacity (hours/day)
  ProcessingTime[i] // Processing time of product i

Output:
Feasible machine-day assignments for all products

Quantum Register Definitions:
  q_var: Machine + Day encoding
  q_sum: qubits for binary sum of processing times
  q_carry: qubits for full-adder carry operations
  q_limit: qubits storing binary value of T_max
  q_ancilla: ancilla qubits for comparison operations
1. Encode each product i into its q_var[i] register
2. Initialize all q_var qubits in uniform superposition with Hadamard gates
3. repeat for r iterations (Grover steps):
4.     // Oracle: Coloring Constraint
5.     for each edge (i, j) in E do
6.       if Machine(i) == Machine(j) then
7.         Mark state as invalid
8.       end if
9.     end for
10.    // Oracle: Machine Capacity Constraint
11.    for each machine m in M and each day d in D do
12.      Use q_sum and q_carry to sum ProcessingTime of all products assigned to (m, d)
13.      Compare sum with q_limit (T_max)
14.      if sum > T_max then
15.        Mark state as invalid
16.      end if
17.    end for
18.    Apply phase flip to all valid states
19.    // Diffusion Operator
20.    Reflect all amplitudes about the mean amplitude
21. Measure q_var registers
22. Decode measured binary string into machine-day assignments
23. Verify constraints classically to confirm feasibility
24. Return best feasible solution
```

4. Simulation and Result

Our trials on real quantum processors were limited to solve graph coloring with 4 and 5 nodes, where each node represents a product with varying processing times. For 4 nodes problem, the processing times were 3 hours, 2 hours, 3 hours, and 4 hours, sequentially. Meanwhile for 5 nodes problem, the processing times were 3 hours, 4 hours, 3 hours, 2 hours, and 4 hours, sequentially. This problem can be written as graph coloring problem let graph G = (V, E) where V is vertices or number of nodes and E is edges. The graphical representation of the 4-nodes and 5-nodes scheduling problems is illustrated in <Figure 4>.



(a) 4-nodes Problem



(b) 5-nodes Problem

Figure 4. Problem Description

Since the quantum circuit required more than 29 qubits, it could not be run on a simulator. Therefore, the constructed circuit was executed on the ibmq_brisbane processor provided by IBM Quantum Experience, with the optimization level set to 0.

4.1 4 Nodes Problem

Let G={{1,2,3,4},{12,13,24,34}), we utilized 34 qubits that were allocated as follows: 12 qubits representing the nodes, 3 qubits for carry operations, 8 qubits for sum calculations, 3 qubits encoding the time limit, and 7 ancilla qubits. The initial state in Grover's algorithm is a uniform superposition over all possible solutions, where each solution corresponds to a distinct binary string from $|000000000000\rangle$ to $|111111111111\rangle$. The total number of possible states N in 12-qubit system is $N = 2^{12} = 4096$.

At the beginning of Grover's search, the algorithm starts by

preparing a uniform superposition of all states. The amplitude of each state in this superposition is equal, and due to the normalization condition, the amplitude α of each state is $\alpha = \frac{1}{\sqrt{N}} = \frac{1}{\sqrt{4096}} = \frac{1}{64}$. After applying the oracle, the amplitudes of the valid states are inverted, while the invalid states remain unchanged. This condition expresses as $\alpha_{afterorade} = \left[\frac{-1}{64}, \frac{1}{64}, \frac{-1}{64}, \cdots, \frac{-1}{64},\right]$.

Following oracle step, the diffusion operator is applied. The diffusion operator amplifies the amplitude of valid states while suppressing the invalid ones. This amplification process works by reflecting the amplitudes of all states about the mean, which results in an increase in the amplitude of valid states and a decrease in the amplitude of invalid states. The calculation of the mean amplitude α can be expressed as:

$$\bar{\alpha} = \frac{\frac{1}{64} + \frac{1}{64} - \frac{1}{64} + \cdots - \frac{1}{64}}{64} \tag{14}$$

Next, we apply the diffusion operator to each amplitude by reflecting it around the mean amplitude $\bar{\alpha}$. After executing the circuit, the result with the highest amplitude was 001010010000, corresponding to a valid solution for the graph coloring problem, as shown in <Figure 5>. This binary string represents the assignment of machines and days for each node, and the associated measurement result frequency was 8. This means that in the sampling process, the quantum state 001010010000 appeared most frequently, indicating that it is the most likely valid solution found by Grover's search.

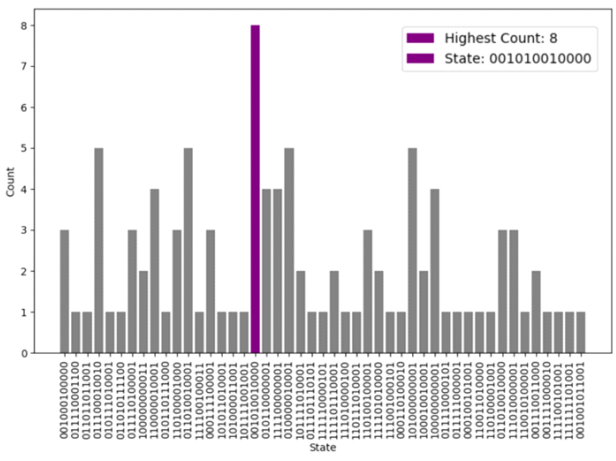


Figure 5. The result of proposed framework for 4 nodes problem after running the circuit on ibmq_brisbane processor provided by the IBM Quantum Experience.

As previously noted, each node is represented by 3 qubits, with the first two qubits indicating the machine and the last qubit indicating the day. Based on this representation, product 1 is allocated to machine 1 on day 2, product 2 to machine 2 on day 1, product 3 to machine 2 on day 1, and product 4 to machine 1 on day 1. The resulting allocation is illustrated in <Figure 6>, providing a visual representation of the product-machine-day assignments.

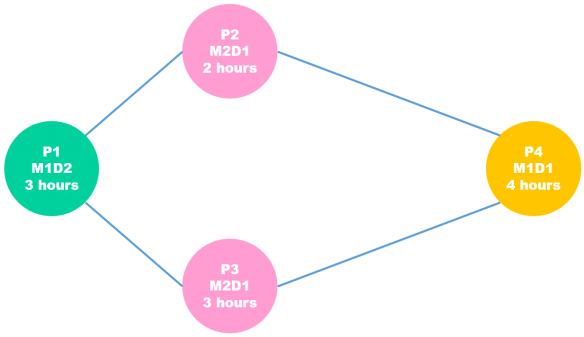


Figure 6. Visual representation of product-machine-day allocation for 4 nodes.

The obtained result satisfies all the problem constraints, ensuring that no two adjacent nodes are assigned to the same machine or the same day and the total processing time for each machine within a single day does not exceed the maximum limit of 6 hours.

4.2 5 Nodes Problem

Let G=({1,2,3,4,5},{12,13,14,24,25,35,45}), we utilized 39 qubits that were allocated as follows: 15 qubits representing the nodes, 3 qubits for carry operations, 12 qubits for sum calculations, 3 qubits encoding the time limit, and 7 ancilla qubits. <Figure 7> presents the results of these executions.

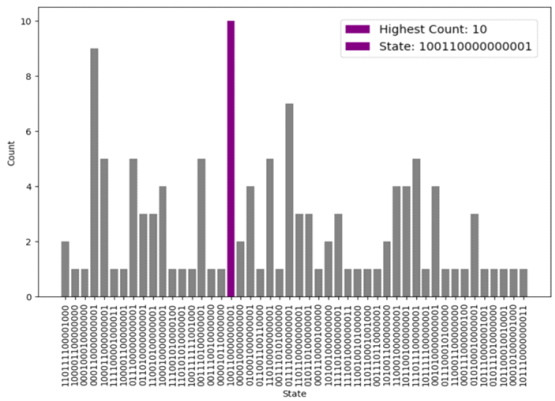


Figure 7. The result of proposed framework for 5 nodes problem after running the circuit on ibmq_brisbane processor provided by the IBM Quantum Experience.

After executing the circuit, the result with the highest amplitude was 100110000000001, corresponding to a valid solution for the graph coloring problem, as shown in <Figure 7>. This binary string represents the assignment of machines and days for each node, and the associated measurement result frequency was 10. This means that in the sampling process, the quantum state 100110000000001 appeared most frequently, indicating that it is the most likely valid solution found by Grover's search. The resulting allocation is illustrated in <Figure 8>, providing a visual representation of the product-machine-day assignments.

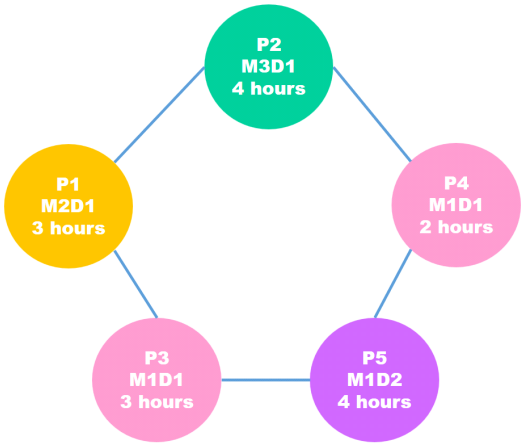


Figure 8. Visual Representation of Product-machine-day Allocation for 5 Nodes

Based on <Figure 7>, product 1 is allocated to machine 2 on day 1, product 2 to machine 3 on day 1, product 3 to machine 1 on day 1, product 4 to machine 1 on day 1 and product 5 to machine 1 on day 2. The obtained result satisfies all the problem constraints, ensuring that no two adjacent nodes are assigned to the same machine or the same day and the total processing time for each machine within a single day does not exceed the maximum limit of 6 hours.

Table 1. Comparison of 4-node and 5-node Graph Coloring Problem

Aspect	4 Nodes	5 Nodes
Possible State	4096	32768
Initial Amplitude	0.015625	0.005525
Time Complexity	$O(64)$	$O(181)$
Circuit Depth	56	80

<Table 1> presents a comparison of 4-node and 5-node graph for solving the graph coloring problem using Grover's algorithm with k=8 colors. The comparison is based on four key aspects: Possible States, Initial Amplitude, Time Complexity, and Circuit Depth. The number of Possible States represents the size of the search space, which increases exponentially with the number of nodes, calculated as k^{nodes}. For the 4 nodes graph, there are 4,096 possible states, while for the 5 nodes graph, the search space expands to 32,768 states. The Initial Amplitude, given by $1/\sqrt{N}$, where N is the number of possible states, is approximately 0.015625 for the 4 nodes graph and decreases to 0.005525 for the 5 nodes graph. This reduction in amplitude reflects the greater difficulty of identifying a solution as the search space grows.

Time Complexity in Grover's algorithm scales proportionally to $O(\sqrt{N})$. Consequently, the time complexity for the 4 nodes graph is $O(64)$, which increases to $O(181)$ for the 5 nodes graph. This aligns with the square root relationship between the number of states and the number of Grover iterations needed. The Circuit Depth, which indicates the number of quantum gates required, also increases from 56 for the 4 nodes graph to 80 for the 5 nodes graph, reflecting the additional computational required to handle a larger problem instance.

4.3 Comparison With Other Quantum Algorithm

To evaluate the performance and suitability of our algorithm for solving the graph coloring problem on quantum processors, two performance metrics are employed, accuracy and success rate. Accuracy quantifies the proportion of measurement outcomes that correspond to feasible schedules satisfying both the graph coloring constraint (no two adjacent products are assigned to the same machine) and the machine capacity constraint (the total processing time allocated to any machine within a single day does not exceed 6 hours). Success rate measures the consistency of the algorithm in producing a valid schedule as its most probable outcome. Specifically, it is computed as the fraction of independent experimental runs in which the highest-frequency measurement result satisfies all problem constraints. It is also essential to compare it with other well-known quantum algorithms, including Grover's algorithm, Quantum Approximate Optimization Algorithm (QAOA), and Quantum Annealing.

The results obtained from solving the graph coloring problem using various quantum algorithms demonstrate distinct performance patterns can be seen in <Figure 9>.

The proposed algorithm consistently achieved higher success rates, with values ranging from 0.35 to 0.75, and an average value around 0.50 to 0.60. The success rates observed in our experiments are consistent with the influence of noise inherent to noisy intermediate-scale quantum (NISQ) processors. The ibmq_brisbane device, on which our framework was executed, is subject to

several dominant error sources, including single-qubit and two-qubit gate infidelities, qubit relaxation, dephasing, and readout errors. These noise processes cumulatively reduce the fidelity of the quantum state, decreasing the amplitude of marked solutions and thereby lowering the measured success probability of Grover's algorithm. This phenomenon aligns with the recent theoretical and experimental findings of Sun (2024), who demonstrated a quantitative complementarity between Grover's success probabilities and decoherence rate. Specifically, the maximum success probability depends on an equation involving the decoherence rate and the total runtime. As noise gets worse (larger decoherence rate), the ability of the algorithm to amplify the correct answer drops, reducing success rates. To mitigate these effects, several strategies were adopted. First, we employed qubit mapping and transpilation at optimization level 0 to minimize unnecessary SWAP operations, thereby reducing circuit depth and exposure to decoherence. Second, measurement error mitigation techniques provided by Qiskit were applied to calibrate readout errors based on the device's current calibration data. Third, we minimized the use of multi-controlled gates by optimizing the oracle and comparator circuits, as such gates are particularly error-prone on NISQ devices.

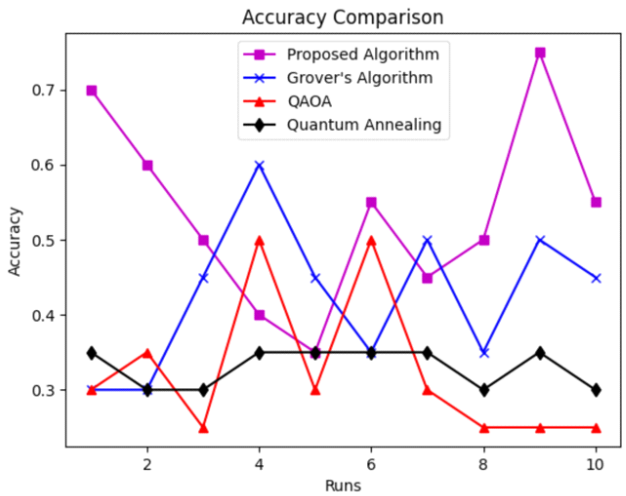


Figure 9. Accuracy Comparison between all Models for 4 Nodes

In comparison, Grover's algorithm displayed moderate success, with results fluctuating between 0.30 and 0.60. While Grover's algorithm performed well in some cases, it was less consistent overall, suggesting that it is effective but not as stable as the proposed method.

On the other hand, the Quantum Approximate Optimization Algorithm (QAOA) exhibited a lower overall performance, with values ranging between 0.25 and 0.50. Finally, quantum annealing produced stable yet modest results, with values tightly between 0.30 and 0.35. While quantum annealing demonstrated reliability, its performance was consistently lower than that of the other algorithms. In terms of solving graph coloring, quantum annealing performs well in satisfying adjacent nodes constraint with optimal value, but this lead fails to satisfy the machine time limitation.

The accuracy comparison for 5 nodes problem can be seen in <Figure 10>. The plot reveals that the Proposed Algorithm consistently achieves higher accuracy than the other methods across multiple runs, often reaching values around 0.5 to 0.7, although it shows some variability. This suggests that while it may be sensitive to noise of real quantum computer, it generally outperforms the other approaches in terms of accuracy. Grover's Algorithm demonstrates moderate variability, typically in the range of 0.3 to 0.5, providing stable results but not reaching the higher accuracy levels seen with the Proposed Algorithm. QAOA (Quantum Approximate Optimization Algorithm) fluctuates significantly, with accuracy values between 0.2 and 0.5. Quantum Annealing, on the other hand, displays the most consistent performance, in the range of 0.3 to 0.4 across all runs. For solving graph coloring, quantum annealing performs effectively in meeting the adjacent nodes constraint with optimal values. However, this success often results in failing to satisfy the machine time limitation.

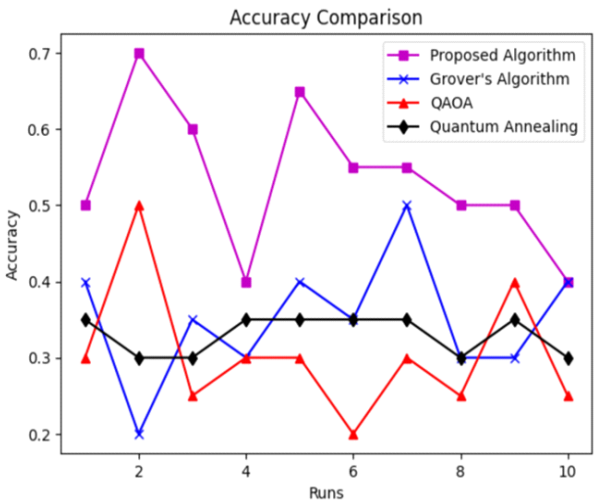


Figure 10. Accuracy Comparison between all models for 5 nodes

In summary, the proposed algorithm demonstrated superior performance and higher accuracy compared to other quantum algorithms, such as Grover's algorithm, the Quantum Approximate Optimization Algorithm (QAOA), and quantum annealing, making it a promising approach for solving the graph coloring problem with additional constraint in quantum computing.

5. Conclusion

This study successfully applied quantum computing, particularly Grover's algorithm with additional constraint, to tackle a constrained scheduling problem modeled as a graph coloring problem. By integrating quantum amplitude amplification, the algorithm efficiently explored large solution spaces to find optimal or near-optimal schedules, adhering to both graph coloring and time constraints. The experiments on a real quantum processor (ibmq_brisbane) demonstrated the algorithm's ability to solve a 4 nodes and 5 nodes graph coloring problem. However, when compared to other quantum algorithms, such as traditional Grover's Algorithm, QAOA, and Quantum Annealing, the solution obtained from proposed framework excelled in providing optimal solutions and satisfying all the constraints.

One of the critical limitations encountered during the trials was the restriction imposed by the current generation of IBM quantum processors, which significantly influenced the scope and complexity of problems that could be addressed. These hardware limitations are common in today's noisy intermediate-scale quantum (NISQ) devices, meaning they can handle only a limited number of qubits and operations.

Moving forward, addressing these hardware constraints will be a key area of focus. Future research will aim to overcome the qubit limitation by optimizing algorithm design and leveraging quantum error correction techniques, which would allow for more stable and reliable computations. Scaling up to solve larger problems such as those involving hundreds of nodes and more complex constraints will require advancements in both hardware, including processors with more qubits and lower noise levels, and algorithms capable of functioning within these quantum limits.

As quantum hardware continues to improve, the ability to solve larger and more complex optimization problems on quantum computers will bring us closer to realizing the full potential of quantum computing in real-world applications such as scheduling and optimization.

References

Ajagekar, A. and You, F. (2019), Quantum computing for energy systems optimization: Challenges and opportunities, *Energy*, **179**, 76-89.

Bravyi, S., Kliesch, A., Koenig, R., and Tang, E. (2022), Hybrid quantum-classical algorithms for approximate graph coloring, *Quantum*, **6**, 678.

Chakrabarty, I., Khan, S., and Singh, V. (2017), Dynamic Grover search: Applications in recommendation systems and optimization problems, *Quantum Information Processing*, **16**, 1-21.

Clarke, J. and Wilhelm, F. K. (2008), Superconducting quantum bits, *Nature*, **453**(7198), 1031-1042.

Coccia, M., Roshani, S., and Mosleh, M. (2024), Evolution of Quantum Computing: Theoretical and Innovation Management Implications for Emerging Quantum Industry, *IEEE Transactions on Engineering Management*, **71**, 2270-2280. https://doi.org/10.1109/TEM.2022.3175633

Do, M., Wang, Z., O'Gorman, B., Venturelli, D., Rieffel, E., and Frank, J. (2020). Planning for compilation of a quantum algorithm for graph coloring. In *ECAI 2020*, IOS Press, 2338-2345.

Farhi, E., Goldstone, J., and Gutmann, S. (2014), A quantum approximate optimization algorithm, *ArXiv Preprint ArXiv:1411.4028*.

Harwood, S., Gambella, C., Trenev, D., Simonetto, A., Bernal, D., and Greenberg, D. (2021), Formulating and solving routing problems on quantum computers, *IEEE Transactions on Quantum Engineering*, **2**, 1-17.

Hong, J. and Lee, H. (2024), Qunatum dense coding transfer framework of phase damping channel system using weak measurement and correction rotation, *Journal of the Korean Institute of Industrial Engineers*, **50**, 202-210.

Khairunissa, M. and Lee, H. (2022), Effective quantum mechanics-embedded nanoparticle occlusion analysis framework, *Journal of Nanoparticle Research*, **24**(7).

Khanal, B., Orduz, J., Rivas, P., and Baker, E. (2023), Supercomputing leverages quantum machine learning and Grover's algorithm, *The Journal of Supercomputing*, **79**(6), 6918-6940.

Lee, H. and Banerjee, A. (2023), Quantum embedding framework of industrial data for quantum deep learning, In *2023 Winter Simulation Conference (WSC)*, 2956-2965. https://doi.org/10.1109/WSC60868.2023.10407584.

Lucas, A. (2014), Ising formulations of many NP problems, *Frontiers in Physics*, **2**, 5.

Mukherjee, S. (2022a), A Grover search-based algorithm for the list coloring problem, *IEEE Transactions on Quantum Engineering*, **3**, 1-8.

Mukherjee, S. (2022b), A Grover search-based algorithm for the list coloring problem, *IEEE Transactions on Quantum Engineering*, **3**, 1-8.

Oh, Y.-H., Mohammadbagherpoor, H., Dreher, P., Singh, A., Yu, X., and Rindos, A. J. (2019), Solving multi-coloring combinatorial optimization problems using hybrid quantum algorithms, *ArXiv Preprint ArXiv:1911.00595*.

Oh, E. and Lee, H. (2024), Quantum mechanics-based deep learning framework considering near-zero variance data, *Applied Intelligence*, **54**(8), 6515-6528.

Oh, E. and Lee, H. (2022), Missing value estimation framework using quantum mechanics-based generative adversarial network, *Journal of Korean Institute of Industrial Engineers*, **32**(6), 457-463.

Perkowski, M. (2022), Inverse problems, constraint satisfaction, reversible logic, invertible logic and Grover quantum oracles for practical problems, *Science of Computer Programming*, **218**, 102775.

Saha, A., Saha, D., and Chakrabarti, A. (2021), Circuit Design fork-Coloring Problem and Its Implementation in Any Dimensional Quantum System, *SNComputer Science*, **2**(6), 427. https://doi.org/10.1007/s42979-021-00813-3.

Stollenwerk, T., Hadfield, S., and Wang, Z. (2020). Toward quantum gate-model heuristics for real-world planning problems, *IEEE Transactions onQuantum Engineering*, **1**, 1-16.

Sun, Y. (2024), Decoherence in Grover search algorithm, *Quantum Information Processing*, **23**(183).
Zhou, L., Wang, S.-T., Choi, S., Pichler, H., and Lukin, M. D.(2020), Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices, *Physical Review X*, **10**(2), 021067.

Author Profile

Lutfiana Sausan: Lutfiana Sausan is a graduate student in School of Industrial Engineering from Kumoh National Institute of Technology from 2024. She received the B.S degree in Industrial Engineering from Muhammadiyah University, Indonesia, in 2023. Her research interests include deep reinforcement learning, optimization and quantum computing.

Hyunsoo Lee: Ph.D. degree in Industrial and Systems Engineering Department at Texas A&M University, College Station, TX, USA in 2010. He received the M.S. degree in Industrial Engineering from POSTECH, Korea in 2002. He is a Professor in School of Industrial Engineering at Kumoh National Institute of Technology, Korea from 2011. His research interests include nonlinear optimization and control, quantum computing, deep learning, and smart manufacturing.