Quantum Information & Computation
ISSN 1533-7146 Volume 25 (2025) pp. 687–715
©Cerebration Science Publishing

Article

Toward an Algorithmic Framework and Grover's search speedup for Quantum Circuit Design—Leveraging the Minimum Connected Dominating Set Problem as an Example

Chu-Fu Wang [1], Yih-Kai Lin [1,*], Chia-Ho Ou [2,3] and Jau-Der Shih [1]

[1] Department of Computer Science and Artificial Intelligence, National Pingtung University, Pingtung, Taiwan 90044, ROC; cfwang@mail.nptu.edu.tw (C.-F.W.); jdshih@mail.nptu.edu.tw (J.-D.S.)
[2] Department of Computer Science and Information Engineering, National Pingtung University, Pingtung, Taiwan 90044, ROC; cho@mail.nptu.edu.tw
[3] Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan
* Corresponding author: yklin@mail.nptu.edu.tw

Received date: 7 September 2025; Accepted date: 20 October 2025; Published online: 25 December 2025

Abstract: In recent years, quantum computing has gradually extended its influence beyond the realm of physics research into the fields of electrical engineering and computer science. Most researchers and programmers remain more familiar with traditional algorithmic techniques based on conventional computer architectures. To address this gap, this study proposes a quantum algorithmic circuit design framework with Grover's search speedup technique and provides theoretical proofs for some design techniques, aiming to facilitate knowledge transfer and ease the learning curve for designers entering the field of quantum algorithm development. Since combinatorial optimization problems in graph theory serve as the foundation for many practical applications, this study adopts the well-known Connected Dominating Set (CDS) problem as a design example to illustrate the practical applicability of the proposed quantum algorithmic design framework and presents a quantum circuit as a potential solution for addressing real-world challenges in network optimization and related applications. In addition, the circuit proposed in this paper can serve as a quantum oracle to identify connected dominating sets (CDSs) of a graph. When the oracle is applied to a superposition of vertex subsets, Grover's search algorithm achieves a quadratic speedup. We designed a method for adjusting the initial amplitudes so that the search can be biased toward smaller CDSs, which maximizes the probability of finding the minimum CDS.

Keywords: index terms-quantum computing; quantum logic; connected dominating set; wireless sensor networks

1. Introduction

Since Peter Shor [1,2] proposed a probabilistic polynomial-time quantum algorithm for integer factorization in 1997, it has inspired many physicists and computer science researchers and has even sparked debates on the possibility of using quantum computing to solve NP-Complete problems. The research on quantum computer and quantum algorithm designs has thus become increasingly popular. The two major features of quantum mechanics are superposition and entanglement. Regarding the quantum superposition property, we can excite multiple quantum bits simultaneously to generate superposition states, thereby expressing various combinations of states, which is akin to generating all possible solutions while solving the optimization problem. However, the difficulty of designing quantum algorithms has also increased significantly compared to traditional algorithms due to the inability to measure the quantum states and their amplitudes during the middle stage of algorithm computation since they will collapse their states after measurement. Another feature, quantum entanglement, allows two entangled quantum particles to maintain their state dependency instantaneously, regardless of the distance between them, giving rise to the new era of quantum communication. This has led to the idea of building a Quantum Internet for the future. The above quantum characteristics open different fashions of methodological development, especially for the quantum algorithm design.

DOI: 10.2478/qic-2025-0037

One can reference to the quantum computing introduction articles [3–5] for most fundamental concepts of quantum computing.

Nowadays, the design of quantum algorithms is divided into two categories: the quantum-logic algorithm design approach [1,2,4,6–11] and the quantum QUBO algorithm design approach [12–17]. Due to the fact that the general-purpose quantum computer is quite primitive and the available number of qubits is generally limited (the IBM company releases its first ever 1,000-qubit quantum chip in 2023), a middle product called the Quantum Ising Machine that is capable of solving optimization problems [13]. The Quantum Ising Machine is a special-purpose quantum computer that performs a quantum annealing algorithm for solving QUBO (Quadratic Unconstraint Binary Optimization) problem. In the case of an optimization problem that can be transformed to a QUBO form, then it can be executed on the Quantum Ising machine (the D-Wave, Toshiba, and Compal Electronics are promising companies that are developing Quantum Quantum Ising machines (or Digital Quantum Ising Machines)). And many applications, such as drug development, scheduling, supply chain, traffic management, etc., have been applied by the quantum QUBO algorithm design approach to solve the optimization problems [15–17]. Literature [12–14] gave a good introduction to this approach. However, the quantum QUBO algorithm design approach only limits the applications to the optimization problem that can be transformed into a QUBO form. For problems that cannot be transformed into the QUBO form, then the quantum Ising machine is not applicable. Thus, quantum-logic algorithm design is the most fundamental approach since these algorithms are designed using existing quantum logic gates (such as the CNOT gate, CCNOT gate, Hadamard gate, NOT gate, etc., [5]) to compose the quantum circuit to perform the computation steps. This approach has fewer limitations compared to the quantum QUBO algorithm design approach. Thus, we will design our proposed algorithm using this approach. Though the quantum-logic algorithm design is the most extensive approach, the design techniques are quite different from the algorithm design in the traditional Von Neumann architecture computer. However, to date, many algorithm designers have been hesitant to adopt the quantum circuit design approach due to unfamiliarity and the high entry barrier associated with current design methodologies. To provide a simplified and accessible framework for algorithm designers could significantly alleviate the current shortage of quantum-logic algorithms. Due to the above reasons, we will introduce a design framework and give the theoretical proofs for some techniques used in the framework for the quantum-logic algorithm design approach. And we will use the CDS as an example to illustrate the applicability of the proposed framework in the later discussions.

The connected dominating set (CDS) problem is defined as follows: A CDS of a graph $G = (V, E)$ is a subset D of V ($D \subseteq V$), such that, (1) every vertex in V either belongs to D or is adjacent to a vertex in D; (2) D induces a connected subgraph of G. The CDS problem aims to find the smallest cardinality among all CDS in G. The CDS problem is known to be an NP-hard problem [18] and many real world applications adopt the solution for the CDS problem as the core set for message relaying in Wireless Sensor Networks [19–23], scheduling for industrial engineering [24], and key generation for encryption [25], etc. The routing for mobile ad-hoc networks or wireless sensor networks is generally challenging. For mobile ad-hoc networks, the network nodes will move frequently, causing the network topology to change dynamically. The CDS will generally play a relatively stable backbone network for message replaying. On the other hand, the nodes in a wireless sensor network are generally equipped with limited battery power. Thus, the CDS will also be a good choice for serving the message relaying route to the sink (see Figure 1). Based on the above-discussed applications for the CDS problem, we believe it is worth proposing a quantum-logic algorithm for solving the CDS problem and also using the proposed design framework to illustrate the solution steps. Literature [10] proposed a biomolecular-inspired quantum algorithm for solving the minimum dominating set problem, which is the close related work for this research. However, the considered problem is different to our research, and the solution approach is also different. The study in [26] presents an efficient implementation of Grover's algorithm for solving the Dominating Set Problem through oracle construction. However, their method does not provide a higher probability of measuring smaller dominating sets. In this work, we address the connected version of the problem. Our contribution is not merely a single solution but rather a framework that systematically constructs quantum circuits for solving graph-theoretic problems. Notably, our approach employs non-uniform initial qubit amplitudes, which effectively enhance the likelihood of measuring smaller connected dominating sets.

The main objectives and contributions of our work are as follows. First, the paper proposes a quantum algorithm design framework intended to help researchers with backgrounds in electrical engineering and computer science transition more smoothly from classical algorithmic design thinking to quantum algorithm design. The framework also incorporates Grover's search method and employs a non-uniform amplitude initialization strategy to accelerate the search for solutions. Second, the framework integrates commonly used techniques in quantum algorithm design, such as qubit value updates. We also provided rigorous mathematical proofs to ensure their correctness. Additionally, the

non-uniform amplitude input for Grover's search is supported by corresponding theoretical validations to ensure performance enhancement. Finally, the paper demonstrates the practical applicability of the proposed approach through the Connected Dominating Set Problem, a representative and widely used problem in graph algorithms, especially relevant in network routing applications.

The organization of this article is as follows: Some basic concepts for quantum computing, the proposed quantum-logic algorithm design framework with accelerated search, and some theoretical proof for the design techniques, will be first introduced in next section. The quantum algorithm and circuit for solving the CDS problem using the proposed quantum algorithmic designed framework and the Grover's search speedup will be illustrated in Section 3 and Section 4, respectively. A simulation result for running the proposed quantum circuit using a numerical example will be demonstrated in Section 5. The concluding remarks will be addressed in the last section.

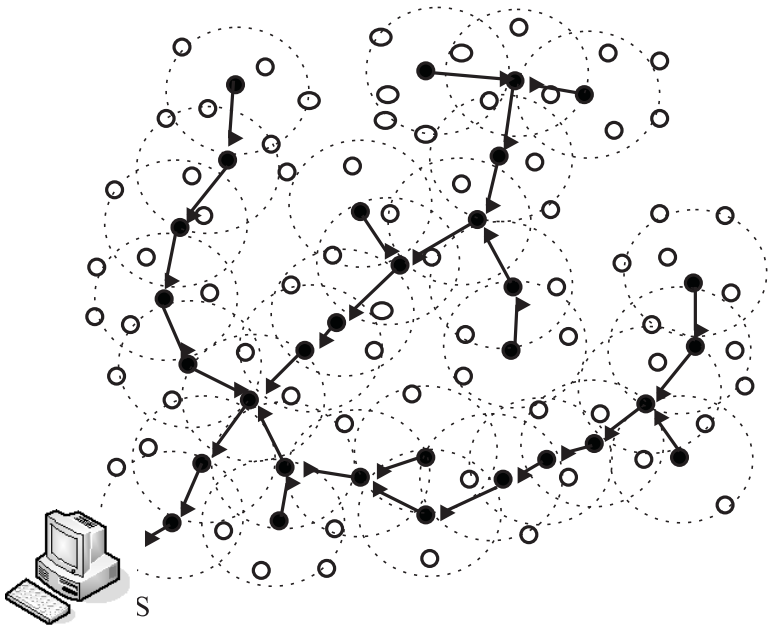


Figure 1. The backbone routing for wireless sensor networks using the connected dominating set.

2. The Proposed Quantum-Logic Algorithm Design Framework with Accelerated Search and Some Preliminary Theoretical Results

2.1. The Proposed Quantum-Logic Algorithm Design Framework and Some Theoretical Results

2.1.1. Characteristic Determination

The quantum mechanism can adopt the photon polarization state to represent values of 0, 1, or more information during the quantum computing process. Generally, the state can be virtually modeled as a unit vector of the Hilbert space. As shown in Figure 2, $|\varphi>= a\cdot|0> +b\cdot|1>$ can be regarded as the combination of states $|0>$ and $|1>$, where values a and b represent the amplitude of the $|0>$ and $|1>$ states, respectively. The values $|a|^2$ (and $|b|^2$) also stands for the probability of the occurring of states $|0>$ (and $|1>$), respectively. Thus, $|a|^2+|b|^2=1$. If $a=1$ and $b=0$ (by letting $\theta=0$), it indicates that the quantum state only exhibits the state $|0>$, and conversely, if $a=0$ and $b=1$ (by letting $\theta=\pi/2$), it signifies that the quantum state only exhibits the state $|1>$. For the special case of letting $\theta=\pi/4$, then $|\varphi>= 1/\sqrt{2}\cdot|0> +1/\sqrt{2}\cdot|1>= 1/\sqrt{2}\cdot(|0> +|1>)$. This is the so called property of superposition of a quantum state, where state $|0>$ and state $|1>$ can coexist with probability $1/2$ of each. The simultaneous representation of $|0>$ and $|1>$ states is a significantly different characteristic from the traditional binary states of 0 or 1 on classical computers. This is the property of superposition inherent in quantum bits (qubits). Based on this property, two qubits can simultaneously express four states: 00, 01, 10, and 11. By extension, multiple qubits can simultaneously express all possible combinations of states.

For ease of representation, quantum mechanism expresses states and operations generally using matrices. For example, the Hardamard gate (H), which generates quantum superposition states, operates on state $|0>$ (vector $(1,0)^T$ representing state $|0>$ and vector $(0,1)^T$ represents state $|1>$) as follows:

$$H(\begin{pmatrix} 1 \\ 0 \end{pmatrix}) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}(\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}) = \frac{1}{\sqrt{2}}(|0> + |1>) \quad (1)$$

In the existing developed quantum logic gates, one primary constraint is imposed by the unitary properties. The unitary of a matrix U, which corresponding to a quantum logic gate, is that the conjugate transpose matrix U^* is its inverse matrix of U; that is, $U^*U = UU^* = I$. Basing on this special requirement, the number of currently developed quantum logic gates is limited, which increase the difficulty of quantum algorithm design.

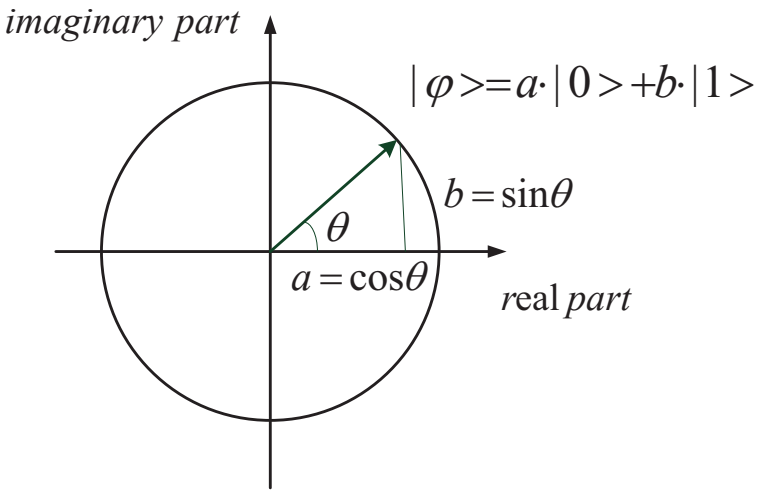


Figure 2. The fundamental concept for quantum states.

2.2. The Proposed Quantum-Logic Algorithm Design Framework and Some Theoretical Results

In the following, we will try to stand on a computer scientist point of view to describe the proposed framework while designing a quantum-logic algorithm using an algorithmic approach.

2.2.1. State Enumeration

For solving an optimization problem, the brute-force method is traditionally used to find the optimal solution by enumerating all the states of the solution space. The superposition property of qubits fits the need to enumerate all possible states of a solution space. For example, as shown in Figure 3(a), the CDS problem in graph $G = (V, E)$ aims to find the minimum size of a connected dominating set D. Thus, the solution space of the considered problem is equal to the collection of all possible subsets of V. For the state enumeration, we use six qubits $|v_i>, 0 \leq i \leq 5$ to represent the vertex selection (or not selection) into a solution. Figure 3(b) shows one of the candidate connected dominating sets ($\{v_1, v_4\}$). In this case, the 6 qubit states are $|v_1>=|v_4>=|1>$ and $|v_0>=|v_2>=|v_3>=|v_5>=|0>$. For ease of discussion, we will use v_i and $|v_i>$ interchangeably to represent the state of a qubit for later discussions. Note that one can adopt the Hardamard gate to apply to each qubit with an initial state of $|0>$ to generate the entire solution space in one quantum operation step (see Figure 3(c)). And we use $|\varphi_{vertex}>=\bigotimes_{i=0}^{|V|-1} H(|v_i = 0>)$ to denote the results (i.e., the entire solution space).

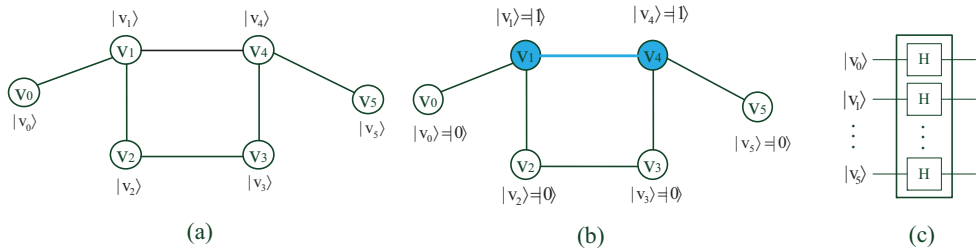


Figure 3. A numerical example of a CDS problem.

2.2.2. **Graph and the Cost Function Representation**

In a graph optimization problem, there will generally be a cost function associated with the vertex set (or the edge set). For example, the vertex cost function values are shown in Figure 4(a). Now one can apply the Pauli-y quantum gate to each qubit to represent the required cost function value on the amplitude of the qubit by the given proper angle θ. Let the cost value that is associated with vertex v_i be $c_i, 0 \leq i \leq |V|-1$. Let the angle be $\theta_i = \sin^{-1}(\sqrt{c_i}), 0 \leq i \leq |V|-1$ and then apply the Pauli-y quantum gate on each qubit using the angle $\theta_i, 0 \leq i \leq |V|-1$. For example, since $c_0 = 0.7$ in Figure 4(a), then $\theta_0 = \sin^{-1}(\sqrt{0.7}) = 56.79^0$. The resulting qubit states will associate with the cost function on the amplitude of state $|1>$. And we use $|\varphi_{vertex}> = \bigotimes_{i=0}^{|V|-1}(\sqrt{c_i}|v_i = 1> + \sqrt{1-c_i}|v_i = 0>)$ to denote the resulting (i.e., the entire solution space) after performing the Pauli-y gate operation on each qubit by given each designate angle $\theta_i, 0 \leq i \leq |V|-1$ (see Figure 4(b)). Besides, the edge qubits set $|\varphi_{edge}> = \bigotimes_{e_{ij} \in E} |e_{ij} = 1>$ can be used to represent a graph topology. For example, the edge qubits set with respect to the graph topology in Figure 4(a) is $|\varphi_{edge}> = \bigotimes\{e_{01} = e_{12} = e_{14} = e_{23} = e_{34} = e_{45} = 1\}$. Note that, we omit the symmetric property here for the undirected graph representation.

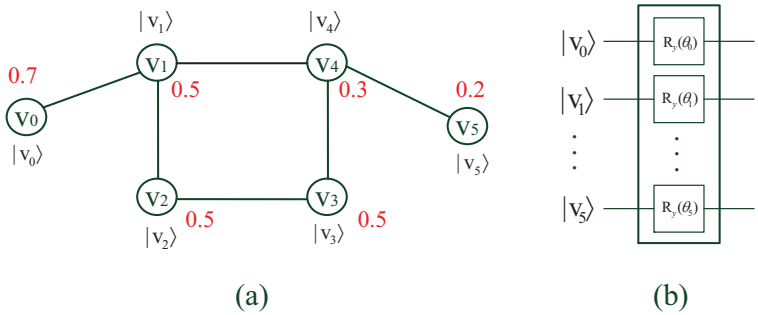


Figure 4. A numerical example for cost function representation.

2.2.3. Characteristic Determination

Generally, the difficult of quantum algorithm design is due to measure the states of quantum bits will cause the states to collapse, and then no more quantum process execution can be done on these quantum bits. In the traditionary algorithm design behavior for the Von Neumann computer, one can easily performe some characteristic determination by Boolean Algebra operations (i.e., using AND, OR, and NOT gates to perform computation for the Disjunctive Normal Form (DNF) formula (or Conjunction Normal Form (CNF) formula). However, the computation for the characteristic determination in quantum computing needs some adjustments.

Firstly, the AND and OR quantum operations (see Figure 5(c)) can be accomplished using the NOT, $CNOT$, and $CCNOT$ (the so-called Toffoli gate) quantum gates (see Figure 5(a)). Note that the intrinsic of the CNOT series of quantum gates (the CNOT, CCNOT, CCCNOT, etc.) is that if the input value of the target line is $|0>$ (or $|1>$), then the result of the target line will be equal to the AND (or NAND) operations made on the values of all control lines (see Figure 5(b)). One can easily check and conclude that, if the value of the control lies is all equal to $|1>$, then the target line will flip the input value (no matter whether the input value is $|0>$ or $|1>$ on the target line). In the rest of this article, for convention, we will regard the results of applying the CNOT series of quantum gates as the determination of the flip (or not flip) of the input value on the target line.

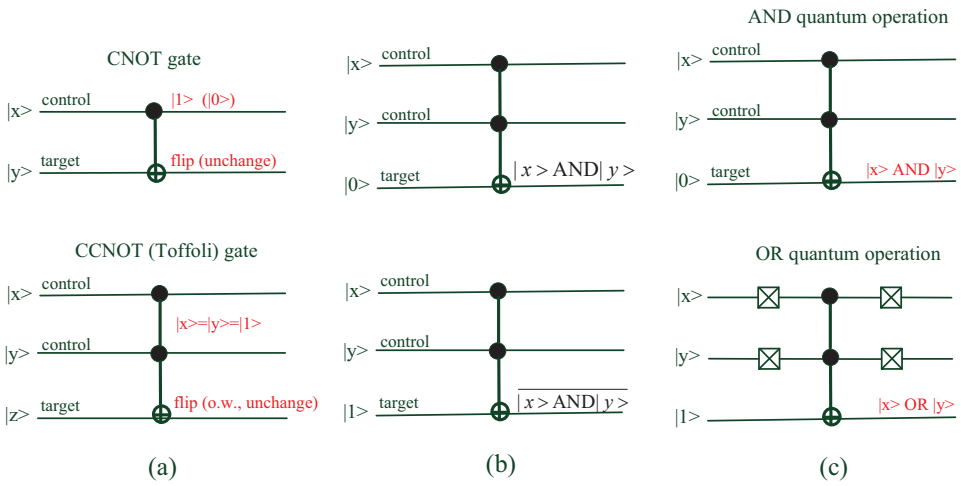


Figure 5. The AND and OR quantum operations.

Though the AND and OR quantum operations can be implemented for quantum operations, one auxiliary flag qubit is required to store the operation result. This kind of flag qubit is generally used to store the computation results of the characteristic determinations. However, in case one would like to update a property determination in one qubit, then a technique is required to perform this task as follows. And we also try to give the technique a theoretical proof as follows.

To generalize this technique, we assume the required task for performing the value update of the formula $x = x + P_1 + P_2 + \ldots + P_n$, where x denotes a quantum bit and $P_i, 1 \leq i \leq n$ is the boolean condition determination, i.e., the result for each $P_i, 1 \leq i \leq n$ is either true/false $(0/1)$. The operator $+$ denotes the OR operation. Note that, $P_i, 1 \leq i \leq n$ may represent the true/false results for the quantum operations that are performed in prior. For ease of illustration, we use the case of $n = 2$ as an example (i.e., the boolean formula is $x + P_1 + P_2$. Since this formula computation would like to keep the result in the qubit x. Thus, the traditional quantum OR and AND operations that shown in Figure 5 no longer fit. To achieve this objective, two steps are performed as follows: At first, the original boolean formula needs to be transformed to its equivalent form by $x + (1 - x) \cdot P_1 + (1 - x) \cdot (1 - P_1) \cdot P_2$, where operator $\cdot$ stands for the AND operation. Then, the equivalent form can be implemented using the $CNOT$, $CCNOT$, $CCCNOT$, ... quantum gates (see Figure 6). We called the above implementation the *qubit formula update*. And to shorten the representation length, we will use $\overline{P_i}$ interchangeably with $1 - P_i, 1 \leq i \leq n$. The generalized correctness proof of the above two steps for implementation of the original boolean formula $x + P_1 + P_2 + \ldots + P_n$ is shown as follows.

$$
\begin{aligned}
& x = x + P_1 + P_2 \\
\rightarrow \quad & = x + \overline{x}P_1 + \overline{x}\,\overline{P_1}P_2
\end{aligned}
$$

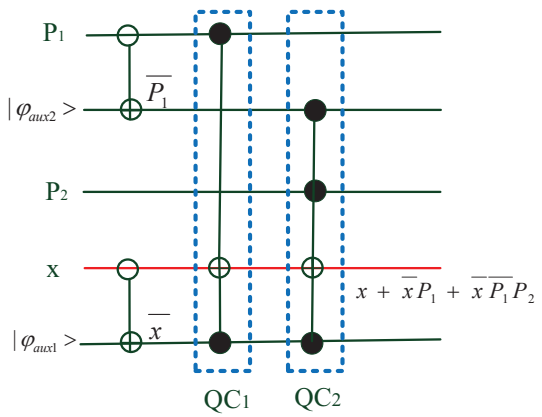


Figure 6. An example for the qubit formula update.

Lemma 1. *The boolean formula* $x + P_1 + P_2 + \ldots + P_{n-2} + P_{n-1} + P_n$ *and the formula* $x + P_1 + P_2 + \cdots + P_{n-2} + P_{n-1} + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}P_{n-1}}P_n)$ *are equivalent for any given true/false values of* $P_i, 1 \leq i \leq n$, *where* $+$ *denotes the OR operation, and* $\cdot$ *denotes the AND operation.*

Proof. Since,

$$\begin{aligned} x + \overline{x}P &= (x + \overline{x})(x + P) \quad \text{(by distributive law of OR operation)} \\ &= 1 \cdot (x + P) \\ &= x + P \end{aligned}$$

Then apply the above result iteratively on each term of P_{n-1}, P_{n-2}, ..., P_2, P_1, we have that,

$$\begin{aligned} & x + P_1 + P_2 + \ldots + P_{n-2} + P_{n-1} + P_n \\ = \; & x + P_1 + P_2 + \ldots + P_{n-2} + P_{n-1} + \overline{P_{n-1}}P_n \text{ (by applying Eq.(2) on the } P_{n-1} \text{ term)} \\ = \; & x + P_1 + P_2 + \ldots + P_{n-2} + P_{n-1} + \overline{P_{n-2}P_{n-1}}P_n \text{ (by applying Eq.(2) on the } P_{n-2} \text{ term)} \\ = \; & \cdots \\ = \; & x + P_1 + P_2 + \ldots + P_{n-2} + P_{n-1} + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}P_{n-1}}P_n) \end{aligned} \tag{2}$$

Hence the lemma. Q.E.D.

□

Lemma 2. *The boolean formula* $x + P_1 + P_2 + \ldots + P_{n-2} + P_{n-1} + P_n$ *and the transformed formula* $x + \overline{x}P_1 + \overline{x}\overline{P_1}P_2 + \ldots + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}P_{n-1}}P_n)$ *are equivalent for any given true/false values of* $P_i, 1 \leq i \leq n$, *where* $+$ *denotes the OR operation, and* $\cdot$ *denotes the AND operation.*

Proof. Applying the result of Lemma 1 iteratively on each term of P_n, P_{n-1}, P_{n-2}, ..., P_1, then we have that,

$$\begin{aligned} & x + P_1 + P_2 + \ldots + P_{n-2} + P_{n-1} + P_n \\ = \; & x + P_1 + P_2 + \cdots + P_{n-2} + P_{n-1} + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}P_{n-1}}P_n) \text{ (by Lemma 1)} \\ = \; & x + P_1 + P_2 + \cdots + P_{n-2} + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}}P_{n-1}) + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}P_{n-1}}P_n) \\ = \; & \cdots \\ = \; & x + \overline{x}P_1 + \overline{x}\overline{P_1}P_2 + \cdots + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}}P_{n-1}) + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-2}P_{n-1}}P_n) \end{aligned} \tag{3}$$

Hence the lemma. Q.E.D.

□

At last, we will try to show that the implementation for the *qubit formula update* will correctly determine the result of the given boolean formula. By Lemma 2, the original boolean formula $x + P_1 + P_2 + \ldots + P_n$ is equivalent to the transformed formula $x + \overline{x}P_1 + \overline{x}\overline{P_1}P_2 + \cdots + (\overline{x}\overline{P_1}P_2 \cdots \overline{P_{n-1}}P_n)$, thus we only need to try to implement the transformed formula using existing quantum gates. According to the quantum circuit design for the *qubit formula update*, we denote the quantum circuit for each subpart of the transformed formula $\overline{x}\overline{P_1}\ \overline{P_2} \cdots \overline{P_{i-1}}P_i), 1 \leq i \leq n$ by QC_i. For example, as shown in Figure 6, QC_1 (and QC_2) are trying to implement the computation for $\overline{x}P_1$ (and $\overline{x}\overline{P_1}P_2$), respectively. Note that, for a higher number of control input of control NOT gate can be easily implemented by some combination of quantum gates, and we will ignore the details here. Now, the following theorem will give the proof of the correction for the *qubit formula update* implementation.

Theorem 1. *The qubit formula update can correctly implement the original boolean formula* $x + P_1 + P_2 + \ldots + P_n$, *for any given true/false values of* $P_i, 1 \leq i \leq n$, *where* $+$ *denotes the OR operation.*

Proof. At first, by Lemma 2, we can focus on the implementation of the transformed boolean formula, since it is equivalent to the original boolean formula. Let the collection of all possible input values for $x, P_1, P_2, \ldots,$ and P_n be

S^* (i.e., $S^* = \{0,1\}^{n+1}$). Let the collection of input values data set that can enable quantum circuit $Q_i, 1 \leq i \leq n$ to flip the results of qubit x be S_i. Since QC_i is designed for implementing the boolean formula subpart $\overline{x}\overline{P_1}\overline{P_2}\cdots\overline{P_{i-1}}P_i$ that is synthesizing by the control NOT series of quantum gates. Thus, $S_i = \{(x, P_1, P_2, \ldots, P_n) \in \{0,1\}^{n+1} | x = 0, \overline{x}\overline{P_1}\overline{P_2}\cdots\overline{P_{i-1}}P_i = 1\}$. Let $S_0 = \{(x, P_1, P_2, \ldots, P_n) \in \{0,1\}^{n+1} \mid x = 1\}$ and $S_{n+1} = \{(x, P_1, P_2, \ldots, P_n) \in \{0,1\}^{n+1} | x = P_1 = P_2 = \ldots = P_n = 0\}$. It is obviously that (1) $\bigcup_{i=0}^{n+1} S_i = S^*$; (2)$S_i \bigcap S_j = \emptyset, \forall i, j \in \{0,1,\ldots,n+1\}$ and $i \neq j$. Note that the first property means that the union of all sets $S_i, 0 \leq i \leq n+1$ will exhaust the entire solution space, and the second property means that these sets are disjoint with each other. And consequently, if the input data causes one of the quantum circuit (say, QC_i) to flip the result, then the other quantum circuits will not flip the result again, which means the total number of flips for the entire sequential operation will be at most once.

Now we will check the transformed formula results by given the input data that comes from set $S_i, 0 \leq i \leq n+1$, and to see whether or not it is equal to the final results of the qubit x. For the case of the input values $(x, P_1, P_2, \ldots, P_n) \in S_0$, then the result for the transformed formula will be equal to 1, and none of the quantum circuits will flip the initial input value of $x = 1$. Thus, the final result of the qubit x will be 1. For the case of the input values $(x, P_1, P_2, \ldots, P_n) \in S_i$, for any $i \in \{1,2,\ldots,n\}$, by definition, since $\overline{x}\overline{P_1}\overline{P_2}\cdots\overline{P_{i-1}}P_i = 1$ the transformed boolean formula will be equal to 1. And since QC_i will cause $x = 0$ to flip once, thus the resulting value will obtain value 1. For the last possible input case of $(x, P_1, P_2, \ldots, P_n) \in S_{n+1}$, obviously the transformed boolean formula will be 0. And since this case of data will not enable any quantum circuit $QC_i, 1 \leq i \leq n$ to flip, thus the result will equal to the original input data, the 0. By the results of Lemma 2 and the above discussions, we have that the result for the original boolean formula will be equal to the result that performing the *qubit formula update*. Hence the theorem. Q.E.D.

□

Now, we will give a numerical example to illustrate the application of this technique. Figure 7(a) gives a single edge graph, that is, $G = (V, E) = (\{v_i, v_j\}, \{e_{ij}\})$. In case one wants to determine whether or not a message can be obtained by vertex v_j. Two vertex qubits v_i and v_j are used to represent the status of whether or not a message can be obtained by vertices v_i and v_j, respectively. A qubit e_{ij} is used to represent the existence of edge e_{ij}. Then the case of vertex v_j will obtain the message that are basing on either one of the following two conditions is met: (1) vertex v_j is already having the message (that is $v_j = 1$) or; (2) vertex v_j's neighbor v_i is having the message (that is $v_i = 1$ and $e_{ij} = 1$). Thus, the boolean formula for updating v_j is $v_j = v_j + v_i e_{ij}$. By the above proposed technique, we have the transformed formula $v_j = v_j + \overline{v_j} v_i e_{ij}$. And the quantum circuit is shown in Figure 7(b). Note that the ancilla qubit $|\varphi_{aux_1}>$ is initialized in the zero state $|0>$.

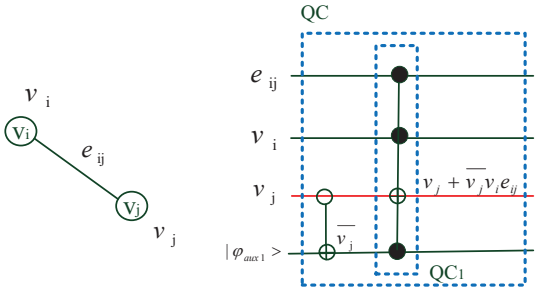


Figure 7. A numerical example for characteristic determination.

2.2.4. Loop Process Representation

Generally, a loop process is frequently used for traditional algorithm design, but it is not provided in quantum-logic algorithm design. Thus, replication is generally a simple technique to implement a loop process. That is, in case a quantum circuit process QC needs to be performed k times, the simplest way is to replicate QC for k times and then synthesize these k quantum circuits QC in sequential order.

2.2.5. Results Output and the Measurement

For solving the combinatorial optimization problem, after generating the entire solution space using the superposition characteristic on the decision variable set (see Figure 8). One may design several quantum circuits (say

$QC_1, QC_2, \ldots, QC_k$) to perform some characteristic determination processes one by one sequentially to obtain the minimum (or maximum) object function values. However, the main objective is to known which solution will result in the optimum solution value. Therefore, the unitary characteristic (that is, $U^*U = UU^* = I$) of the quantum circuit can reverse the operation to obtain the original decision variable values (that is, the optimum solution). Thus, the right part of the quantum circuit in Figure 8 can achieve this objective.

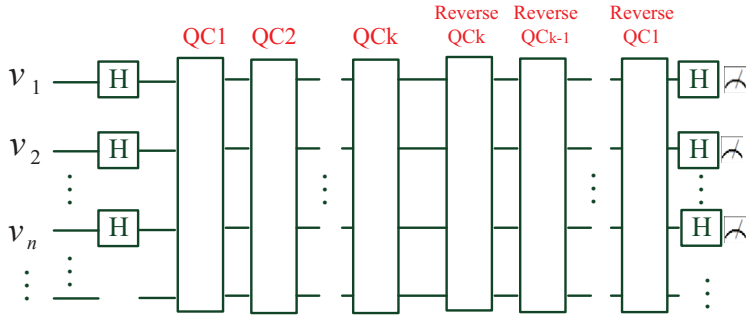


Figure 8. An example for results output and the measurement.

2.2.6. Circuit as an Oracle for Accelerated Search

The quantum circuit we designed can also act as an oracle to speed up the search. The oracle encodes the problem structure so that it can mark which states are valid solutions and which are not. For example, if the input qubit state represents a subset of vertices, the oracle checks whether this subset is a connected dominating set (CDS). If the input is a superposition of all possible subsets, then Grover's algorithm can be applied to find a CDS with a quadratic speedup of $O(\sqrt{2^n})$, where n is the number of vertices. Measuring the final state will then give a bitstring that corresponds to a CDS.

In addition, the initial amplitudes of the superposition can be adjusted so that smaller CDSs have a higher probability of being measured. After running Grover's iterations, the algorithm is therefore more likely to output a smaller CDS. In this way, the oracle not only reduces the search complexity compared to classical methods, but also helps direct the quantum search toward the most desirable solutions.

3. The Solution Method for Solving the Connected Dominating Set Problem Using the Proposed Design Framework

The complete architecture for the proposed method to solve the CDS problem is shown in Figure 9. As shown, the proposed method is composed of the following 5 processes sequentially: the *state enumeration*, the *dominating test process*, the *connected test process*, the *size computation process*, and the *results output & measurement*. In the following, we will illustrate these processes in detail using the algorithmic description approach.

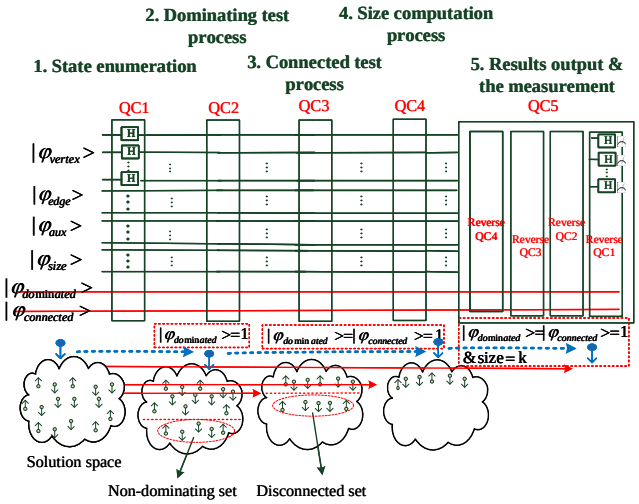


Figure 9. The architecture of the proposed solution method.

3.1. State Enumeration

In this process, the graph input and the state space generation follow the above-mentioned technique. Therefore, given a graph topology $G = (V, E)$ for the considered problem, we use qubits $|\varphi_{edge}>= \bigotimes_{e_{ij}\in E} |e_{ij} = 1>$ to denote the resulting graph topology, and let qubits $|\varphi_{vertex}>$ to represent the vertex set. We then apply the Hardamard gate to each vertex qubit with an initial value of 0, to obtain the entire solution space of the considered problem; that is, $\bigotimes_{i=0}^{|V|-1} H(|v_i = 0>)$. We denote this quantum circuit for the state enumeration process as QC_1. Note that some auxiliary qubits and characteristic determination qubits are also added to this quantum circuit QC_1, which will be used for later quantum computations.

3.2. Dominating Test Process

In this process, we add $|V|$ auxiliary qubits $|\varphi_{\hat{vertex}}>= \bigotimes_{i=0}^{|V|-1} |\hat{v}_i = 0>$ to be the dominated flag with respect to each vertex $v_i \in V$, for representing whether or not the vertex is dominated. According to the definition of a vertex $v_i \in V$ being dominated, it is to test whether or not one of the following two conditions is met: (1) vertex v_i belongs to the dominating set (the P_1 clause); (2) there exists a neighbor of v_i that belongs to the dominating set (the P_2 clause). Thus, by the above proposed technique, the dominated flag qubit $\hat{v}_i$ with respect to vertex v_i can be updated by $\hat{v}_i = \hat{v}_i + \bar{\hat{v}}_i v_i + \bar{\hat{v}}_i \bar{v}_i v_j e_{ij}$, and the quantum circuit is denoted by $QC2_{ij}$. Since the process on each edge $(v_i, v_j) \in E$ must be performed bidirectionally, each edge must perform the quantum operations of $QC2_{ij}$ and $QC2_{ji}$, respectively. As shown in Figure 10(a), after performing the state enumeration process, the entire solution space will be generated. Then any solution (for example, the $|\varphi_{vertex}>= |010010>$, which stands for the candidate vertex set $\{v_1, v_4\}$), will perform the above quantum operations (see Figure 10(b)). The complete operations for updating all dominated flags must be performed on each edge of the given graph. Thus, a loop is required for the operations. Figure 11(a) shows the loop implementation of the numerical example.

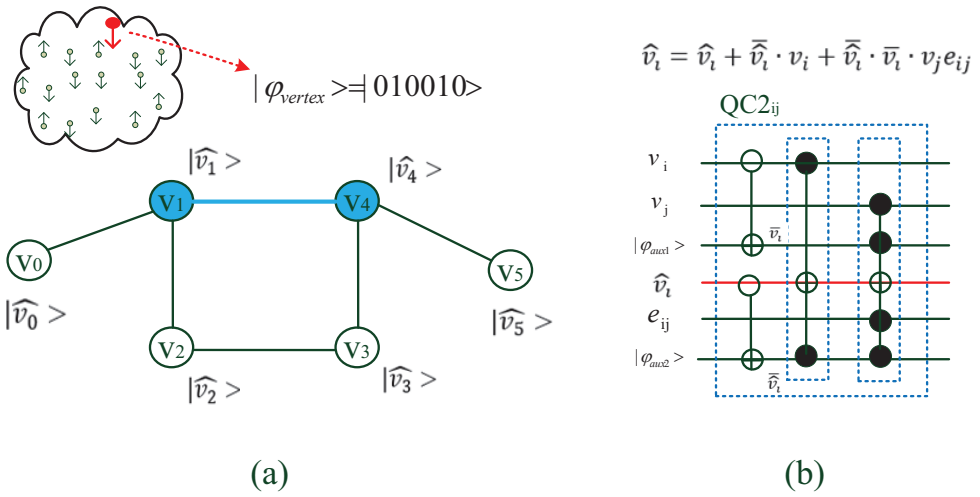


Figure 10. The solution method for the dominating test process (the dominated flag qubits updating operations).

Now, we need to determine whether this solution is a dominating set or not, and store the result into the characteristic determination qubit $|\varphi_{dominated} = 0>$. Note that if all of the dominated flag qubits $\hat{v}_i = 1, \forall v_i \in V$, then the characteristic determination qubit will be $|\varphi_{dominated} = 1>$, which means the solution is a dominating set. For the rest of the cases the qubit remains $|\varphi_{dominated} = 0>$, which means that the solution is not a dominating set. Since we need to test whether every dominated flag qubit is equal to 1 or not, it can be easily implemented using CCCNOT quantum gates, and we use $QC2\_b$ to denote the quantum circuit. Figure 11(b) gives a general description of the implementation. In the illustration, we only give a simple description to determine the result. In general, one can try to modify the quantum circuit design by reusing the auxiliary qubits, and we omit them here.

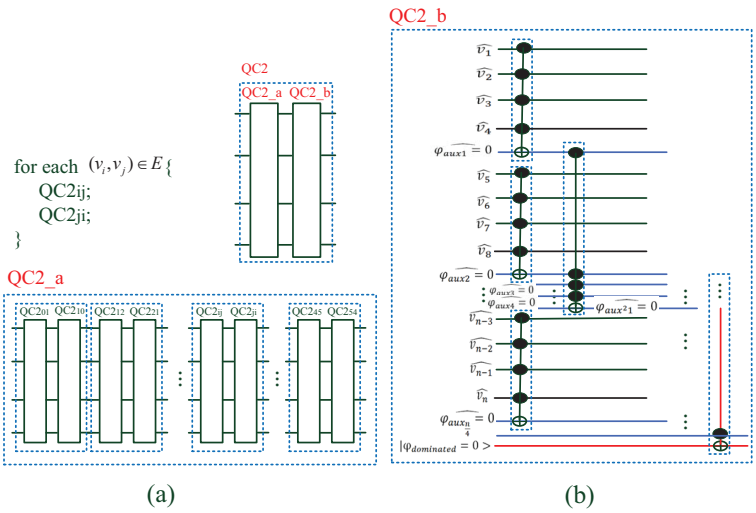


Figure 11. The solution method for the dominating test process (the dominating test qubit characteristic determination operations).

3.3. Connected Test Process

In this process, the operations of the quantum circuit ($QC3$) will determine whether a candidate solution ($|\varphi_{vertex}>$) is connected or not, and then it stores the true/false result into the quantum bit $|\varphi_{connected}>$. At first, we propose a subprocess called *Reachability testing* to test whether or not all of the vertices in the candidate solution are reachable from a given seed vertex $v_k \in V$ and the true/false result will be stored in an auxiliary qubit NOT-pass^k. Note that if the seed vertex $v_k \in V$ belongs to the candidate solution, then the *Reachability testing* is an effective operation. If the resulting auxiliary qubit NOT-pass$^k = 0$ (or 1), then it means the candidate solution is connected (or disconnected). On the contrary, if the seed vertex $v_k \in V$ does not belong to the candidate solution, then the *Reachability testing* is an ineffective process, and the resulting auxiliary qubit will obtain NOT-pass$^k = 1$. Since one cannot guarantee that the planting seed vertex belongs to the candidate solution, the *Connected test process* will invoke the *Reachability testing* for $n(= |V|)$ times and one for letting any vertex $v_k \in V$ as a seed vertex. In case the auxiliary qubits NOT-pass$^k, v_k \in V$ returns a value of 0, then we can conclude that the candidate solution is connected; otherwise, it is disconnected.

The main operations for the subprocess *Reachability testing* are as follows: A reachable flag qubit $\hat{v}_i$ is associated with each vertex $v_i \in V$ and is initially set to be 0. Note that since these qubits $|\varphi_{vertex}>$ can be reused with the dominated flag qubits in the above *Dominating test process*, we used the same notation here ($\hat{v}_i, \forall v_i \in V$). Each time a vertex (say $v_k \in V$) is chosen to be a seed vertex, then the corresponding reachable flag qubit will be set to be $\hat{v}_k = 1$. The reachable characteristic can propagate from a vertex $v_i \in V$ to its neighbors $v_j \in V$ according to the following two either-or conditions: (1) either vertex v_j is already reachable from the seed vertex (that is $\hat{v}_j = 1$); or (2) vertex v_i is reachable from the seed vertex ($\hat{v}_i = 1$), vertex v_j belongs to the candidate solution ($v_j = 1$), and vertex v_j is a neighbor of v_i ($e_{ij} = 1$). Again, one can easily apply the above-mentioned technique to perform the updated formula ($\hat{v}_j = \hat{v}_j + \bar{\hat{v}}_j \cdot \hat{v}_i v_j e_{ij}$) on each edge e_{ij}. The corresponding quantum circuit is shown in Figure 12(b), and we denote this quantum circuit as $QC3_{ij}$.

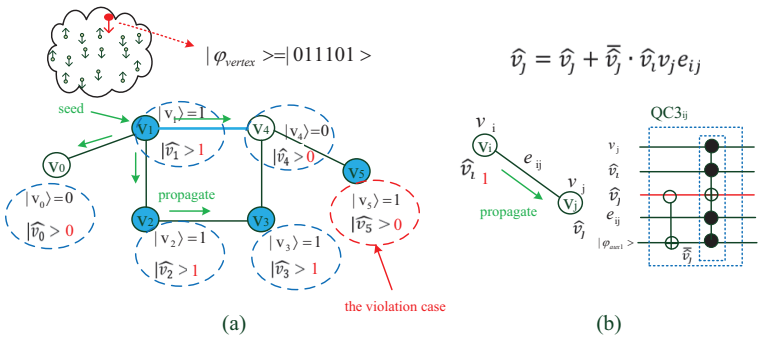


Figure 12. The solution method for the *Connected test process* (the reachable flag qubit updating operations).

Before performing the *Reachability testing* subprocess, we will set the reachable flag qubits $|\varphi_{ver\hat{t}ex} >= \otimes_{v_i \in V} |\hat{v}_i = 0 >$ for initialization. We then pick a seed vertex v_k for planting the value 1 into its reachable flag qubit (that is, $\hat{v_k} = 1$). Then an expanding operation is applied to perform a double loop on each vertex in $v_i \in V$ with its adjacent edge $e_{ij} \in E$ to propagate the reachable characteristic as much as it can (that is, the performing of the quantum operations $QC3_{ij}$ and $QC3_{ji}$ with respect to edge $e_{ij} \in E$). The quantum circuit with respect to the expanding operations of the *Reachability testing* process is denoted by $QC3^k_{\text{Reach-a}}$ (see Figure 13). As shown in Figure 13, the red dashed rectangle of the pseudo code is equivalent to the red dashed rectangle of the quantum circuit, which is the implementation of the numerical example shown in the upper right of Figure 13.

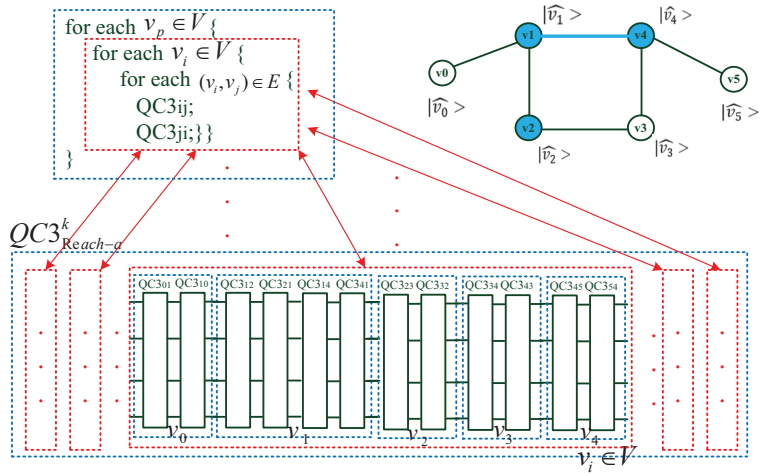


Figure 13. The quantum operations for the *Connected test process* (the Reachability test operations ($QC3^k_{\text{Reach-a}}$)).

The above expanding operation will let the seed of the initial reachable flag value propagate as much as it can. If a seed is planted into a vertex that belongs to the candidate solution vertex set and the solution set is connected, then both of the qubit values of v_i and $\hat{v}_i$ will be equal to 1, for every vertex v in the candidate solution. We will say that every vertex in the candidate solution is reachable, which means that the candidate solution is connected, and the auxiliary qubit will obtain the result of NOT-pass$^k = 0$. On the contrary, some violations will occur in the case of the qubit $v_i = 1$, but the vertex is not reached by the expanding operation (that is, the reachable flag qubit of this vertex is $\hat{v}_i = 0$) (see Figure 12(a)). Of course, the auxiliary qubit will obtain the result of NOT-pass$^k = 1$. In order to obtain the above-mentioned results, one can apply the following updated formula to each vertex $v_i \in V$, NOT-pass^k = NOT-pass^k + NOT-pass$^k \cdot v_i \bar{\hat{v}}_i$ (the corresponding quantum circuit is shown in Figure 14(a) and (b)). Lastly, combining the two quantum circuits $QC3^k_{\text{Reach-a}}$ and $QC3^k_{\text{Reach-b}}$, then the implementation for the *Reachability testing* process is obtained, and we denoted it by $QC3^k_{\text{Reach}}$ (see Figure 14(c)). The detailed operations of the *Reachability testing* process are shown in Figure 15.

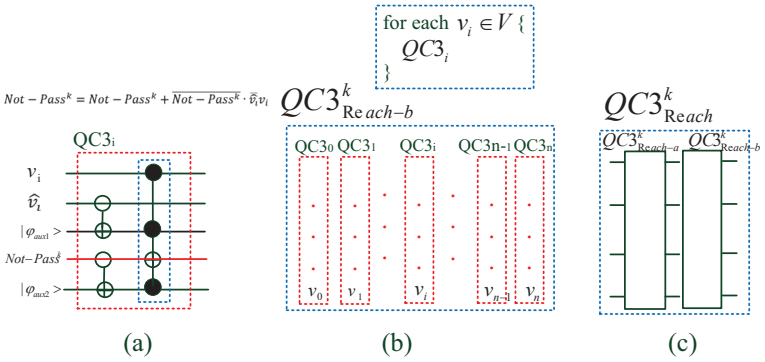


Figure 14. The quantum operations for the *Connected test process* (the Reachability test operations ($QC3^k_{\text{Reach-b}}$ and $QC3^k_{\text{Reach}}$)).

Process Reachability-testing();
(1) // Reachable characteristic expanding operation ($QC3^k_{\text{Reach-a}}$)
(2) **for** each ($v_p \in V$)
(3) **for** each ($v_i \in V$)
(4) **for** each neighbor edge of v_i ($e_{ij} \in E$)
(5) // perform the updated formula $\hat{v}_j = \hat{v}_j + \bar{\hat{v}}_j \cdot \hat{v}_i v_j e_{ij}$;
(6) $QC3_{ij}$;
(7) $QC3_{ji}$;
(8) // Reachable characteristic testing operations ($QC3^k_{\text{Reach-b}}$)
(9) NOT-pass$^k = 0$;
(10) **for** each ($v_i \in V$)
(11) // perform the updated formula NOT-pass^k = NOT-pass^k + $\overline{\text{NOT-pass}}^k \cdot v_i \bar{\hat{v}}_i$;
(12) $QC3_i$;

Figure 15. The operation steps for the Reachability testing process.

Now we will illustrate the complete operations for the *Connected test process* as follows: Initially, we let the characteristic determination qubit $|\varphi_{connected}>$ be 0. Since we do not know whether the seed will be planted into a vertex in the candidate solution set or not, a loop will be performed for the seed being planted into each vertex $v_k \in V$. When a seed is planted into a vertex $v_k \in V$, an auxiliary qubit NOT-pass^k is used to store the result of this *Reachability testing process* and we set it to 0 and let $\hat{v}_k = 1$ for initialization. After performing the *Reachability testing process*, if the candidate solution $|\varphi_{vertex}>$ is connected, then one of the *Reachability testing subprocesses* for the loop will return a successful result; that is, the auxiliary qubit NOT-pass$^k = 0, \exists v_k \in V$. Otherwise, none of the *Reachability testing subprocesses* for the loop will return a successful result. Again, the characteristic determination qubit $|\varphi_{connected}>$ can perform the following updated formula: $\varphi_{connected} = \varphi_{connected} + \varphi_{\overline{connected}} \cdot \overline{\text{NOT-pass}}^k \cdot v_k$, for each time of *Reachability testing subprocess* for the seed vertex to be $v_k \in V$. Note that the term v_i in the updated formula is to ensure that if the seed is planted into a non-solution set vertex (that is, $v_i = 0$), then the *Reachability testing subprocess* will fail. The complete operations of the *Connected test process* are shown in Figure 16.

Process Connected-test();
(1) // The seed being planted into each vertex $v_k \in V$
(2) $\varphi_{connected} = 0$;
(3) **for** each ($v_k \in V$)
(4) **if** ($v_k == 1$) then $\hat{v}_k = 1$; else $\hat{v}_k = 0$;
(5) // perform the *Reachability testing process* and the updated formula
(6) // $\varphi_{connected} = \varphi_{connected} + \varphi_{\overline{connected}} \cdot \overline{\text{NOT-pass}}^k \cdot v_k$
(7) $QC3^k_{\text{Result-integrate}}$;

Figure 16. The operation steps for the *Connected test process*.

The corresponding quantum circuit for the *Connected test process* is shown in Figure 17. Note that, since the quantum gate operation cannot directly implement the if-then-clause of the seed planted process in Figure 16, we will use the CNOT quantum gate to apply it to the control line qubit $v_k \in V$ and let the target line be the qubit $\hat{v}_k = 0$. If the seed is being planted into a non-candidate solution vertex $v_k \in V$, then the resulting $\hat{v}_k$ will remain unchanged (that is, $\hat{v}_k = 0$). In such a case, the resulting auxiliary qubit will obtain a fail result (that is, NOT-pass$^k = 1$). Otherwise, if the seed is planted into a candidate solution vertex $v_k \in V$ (that is $v_k = 1$), then the resulting auxiliary qubit NOT-pass^k will follow the *Reachability testing process* result and then be integrated by the updated formula: $\varphi_{connected} = \varphi_{connected} + \varphi_{\overline{connected}} \cdot \overline{\text{NOT-pass}}^k \cdot v_k$ (see Figure 17(a)). The complete quantum circuit ($QC3$) for the *Connected test process* is shown in Figure 17(b).

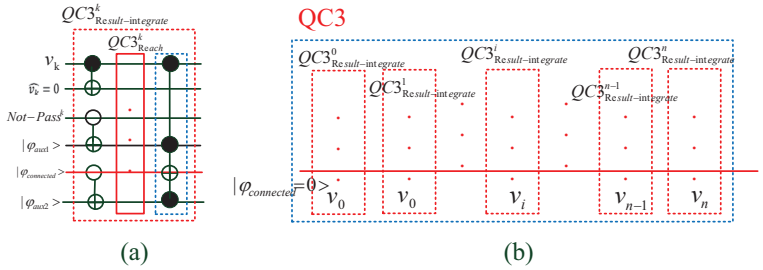


Figure 17. The quantum circuit for the *Connected test process* ($QC3$).

3.4. Size Computation Process

In this process, the size of a solution set $|\varphi_{vertex}>$ will be determined, and the result will be stored into an auxiliary qubit set $|\varphi_{size}>$ using the binary numeral system representation. For the example shown in Figure 13, the size of the candidate solution set $|\varphi_{vertex}>=|011010>$ is equal to 3; thus, $|\varphi_{size}>=|s_2s_1s_0>=|011>$. Obviously, the number of auxiliary qubits required to store the result is equal to $\lceil \log_2 n \rceil + 1$, where n stands for the number of vertices (that is, $|V|$). In the following, two approaches (the *Half adder approach* and the *Toffoli gate approach*) are proposed to achieve the above objective. Detailed descriptions are illustrated using the numerical example of $|\varphi_{vertex}>=|v_0v_1v_2v_3>$ and $|\varphi_{size}>=|s_2s_1s_0>$. For any input graph size n can be easily generalized, so we omit it here.

3.5. Half Adder Approach

A half adder is quite simple for traditional digital logic. It can add two digits (a and b), and produce a sum bit (S) and a carry bit (C). The carry bit C (and the sum bit S) is equal to the result of performing operations a AND b (and a XOR b), respectively. The quantum circuit with respect to a half adder can be easily implemented (see Figure 18(a)). Now, the main idea for the Half adder approach is to perform the addition for each group of two vertices (say, v_i and v_{i+1}), which are shown in the first column of the quantum circuit in Figure 18(b). Note that the carry bits of a half adder are the weight of 2^1. If we continue to apply a half adder to both of the carry bits of two half adders, the weight of the resulting carry bit will be equal to 2^2. Following this principle and taking the weight of each output into consideration, the quantum circuit with respect to the *Size computation process* can then be built. Figure 18(b) gives a quantum circuit ($QC4$) for a graph with order 4 (that is, $|\varphi_{vertex}>=|v_0v_1v_2v_3>$ and the auxiliary qubits set is $|\varphi_{size}>=|s_2s_1s_0>$). In this numerical example, if $|\varphi_{vertex}>=|v_0v_1v_2v_3>=|1111>$. As shown in Figure 18(b), one can easily check and obtain the result $|\varphi_{size}>=|s_2s_1s_0>=|100>$.

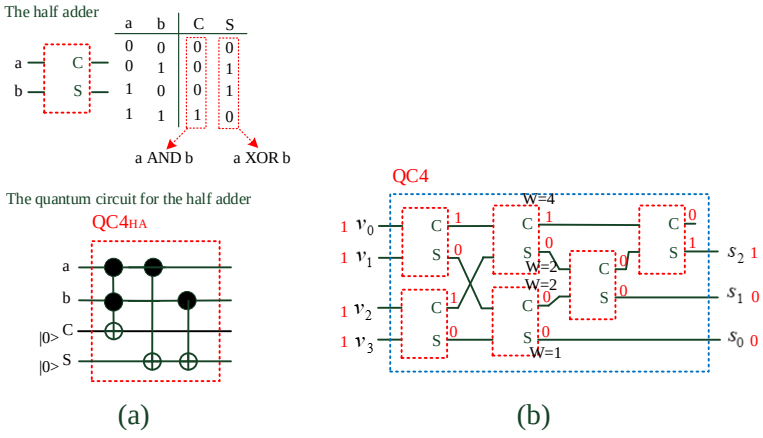
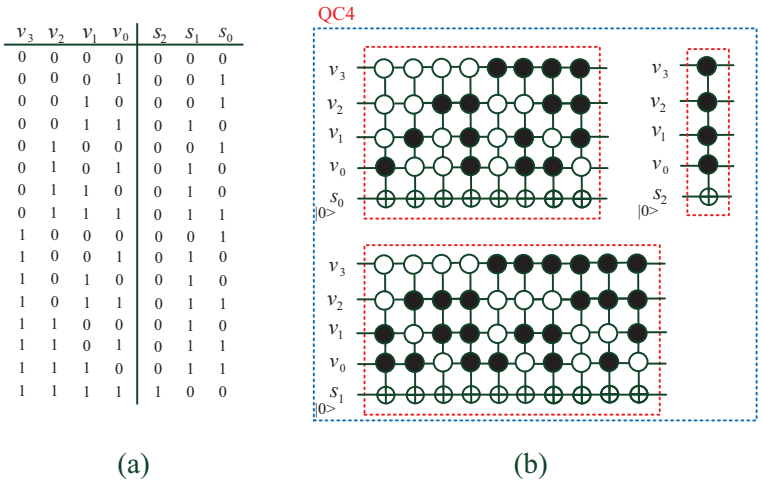


Figure 18. The quantum circuit for the *Size computation process* ($QC4$) (the Half adder approach).

3.6. Toffoli Gate Approach

This approach first builds a truth table for the size determination results. We use the same numerical example that was used for the Half adder approach (see Figure 19(a)). Then, for each resulting auxiliary qubit, s_0, s_1, and s_2 can be emulated using a series of Toffoli gates (see Figure 19(b)). Detailed descriptions are omitted here.



v_3	v_2	v_1	v_0	s_2	s_1	s_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	0	0

Figure 19. The quantum circuit for the *Size computation process* ($QC4$) (the Toffoli gate approach).

3.7. Results Output & the Measurement

Now, we can apply the quantum-logic algorithm design technique discussed in Section 2 to build the final stage of the quantum circuit for our proposed method. After that, one can start to measure the minimum connected dominating set by giving the following conditions: $|\varphi_{dominated} = 1>$, $|\varphi_{connected} = 1>$, and $|\varphi_{size} = 001>$ (see Figure 20). Note that if the measure returns that the solution is empty, then the size value can increase by 1 and try to measure the result again until the solution is not empty.

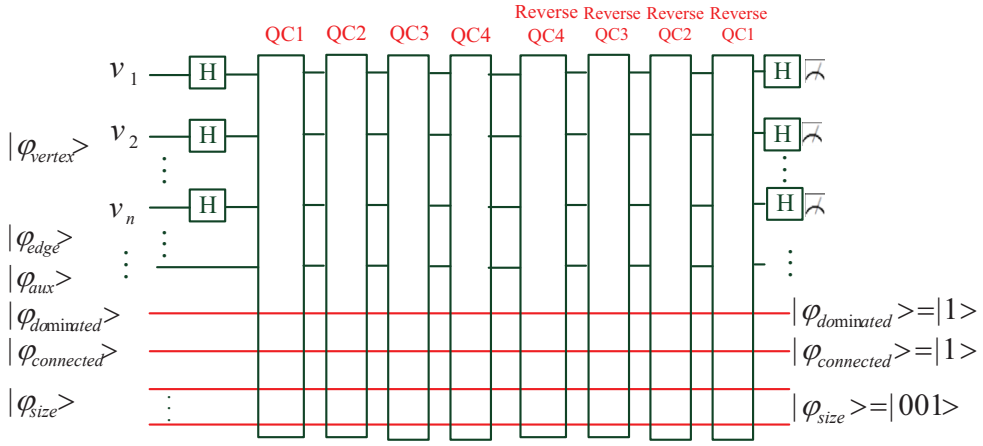


Figure 20. The illustration of the *Results output & the measurement*.

4. Using the Proposed Method as the Oracle in Grover's Algorithm for Minimum Connected Dominating Set Search

The circuit proposed in Section 3 can be used to determine whether an input qubit state, representing a vertex subset, is a connected dominating set of a given graph, as illustrated in Figure 9. If the input to the circuit is not a single subset of vertices but rather a superposition of all possible subsets of vertices, we can apply Grover's search algorithm to find a connected dominating set, achieving a speed-up of $\sqrt{2^n}$, where n is the number of vertices. In practice, the circuit described in Section 3 serves as the oracle in Grover's algorithm. A measurement of the final quantum state yields a bitstring corresponding to a connected dominating set of the graph.

Let the set of all connected dominating sets be $S_1, S_2, \ldots, S_M$, where each S_i is a subset of $v_1, v_2, \ldots, v_n$ and M is the number of solutions. As the primary objective of this paper is to find the minimum connected dominating set, the ideal outcome is one where the measurement probability of obtaining S_i is higher than that of S_j if $|S_i| < |S_j|$. In other words, when measuring the result after running Grover's algorithm with the proposed circuit as the oracle, a smaller CDS is more likely to be obtained than a larger one.

To achieve this objective, the initial superposition is prepared with non-uniform amplitudes. Let n be the number of qubits, $N = 2^n$ the dimension of the Hilbert space, and $\{|x\rangle\}_{x=0}^{N-1}$ the computational basis, where $x \in \{0,1\}^n$. Let $s_2(x)$ denote the number of ones in the binary representation of x. The weight of the bitstring x is defined as

$$w(x) = n - s_2(x),$$

which corresponds to the number of zeros in the bitstring x. These scores are collected into a vector $\mathbf{w} = \begin{bmatrix} w(0) & w(1) & \dots & w(N-1) \end{bmatrix}^\top \in \mathbb{R}^N$. To obtain the amplitude coefficients, this vector is normalized:

$$\alpha_x = \frac{w(x)}{\|\mathbf{w}\|}, \quad \text{where } \|\mathbf{w}\| = \sqrt{\sum_{x=0}^{N-1} w(x)^2}.$$

Let $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_0 & \alpha_1 & \dots & \alpha_{N-1} \end{bmatrix}^\top$ be the resulting normalized amplitude vector. The initial state $|v\rangle$ for the n qubits of the circuit proposed in Section 3 is then prepared as:

$$|v\rangle = \sum_{x=0}^{N-1} \alpha_x |x\rangle .$$

By definition, the computational basis $\{|x\rangle\}_{x=0}^{N-1}$ is orthonormal. That is, for any target state $|x_t\rangle \in \{|x\rangle\}_{x=0}^{N-1}$, we have $\| |x_t\rangle \|^2 = \langle x_t | x_t \rangle = 1$. We also show that $|v\rangle$ is a unit vector. First, the amplitudes: $\sum_{x=0}^{N-1} |\alpha_x|^2 = \frac{1}{\|\mathbf{w}\|^2} \sum_{x=0}^{N-1} w(x)^2 = 1$. The squared norm of the state $|v\rangle$ is calculated by leveraging the orthonormality of the computational basis $\{|x\rangle\}_{x=0}^{N-1}$:

$$\begin{aligned} \| |v\rangle \|^2 &= \langle v | v \rangle \\ &= \langle \sum_{y=0}^{N-1} \alpha_y |y\rangle \, | \sum_{x=0}^{N-1} \alpha_x |x\rangle \rangle \\ &= \sum_{x,y=0}^{N-1} \alpha_y^* \alpha_x \langle y | x \rangle \\ &= \sum_{x=0}^{N-1} |\alpha_x|^2 \\ &= 1. \end{aligned}$$

Thus, $|v\rangle$ is a properly normalized quantum state (i.e., a unit vector).

For example, let $n = 2$, so the computational basis is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. For each $x \in \{0,1,2,3\}$, we calculate: $s_2(0) = 0$, so $w(0) = 2 - 0 = 2$, $s_2(1) = 1$, so $w(1) = 2 - 1 = 1$, $s_2(2) = 1$, so $w(2) = 2 - 1 = 1$, and $s_2(3) = 2$, so $w(3) = 2 - 2 = 0$. Thus, the weight vector is: $\mathbf{w} = \begin{bmatrix} 2 & 1 & 1 & 0 \end{bmatrix}^\top$. The norm is computed as: $\|\mathbf{w}\| = \sqrt{2^2 + 1^2 + 1^2 + 0^2} = \sqrt{4 + 1 + 1} = \sqrt{6}$. Therefore, the normalized amplitude vector is: $\boldsymbol{\alpha} = \frac{1}{\sqrt{6}} \begin{bmatrix} 2 & 1 & 1 & 0 \end{bmatrix}^\top$. The initialized quantum state is: $|v\rangle = \frac{2}{\sqrt{6}} |00\rangle + \frac{1}{\sqrt{6}} |01\rangle + \frac{1}{\sqrt{6}} |10\rangle + 0 \cdot |11\rangle$. To verify that the norm of the state $|v\rangle$ is equal to 1: $\| |v\rangle \| = \sqrt{\langle v | v \rangle} = \sqrt{|\frac{2}{\sqrt{6}}|^2 + |\frac{1}{\sqrt{6}}|^2 + |\frac{1}{\sqrt{6}}|^2 + |0|^2} = \sqrt{\frac{4}{6} + \frac{1}{6} + \frac{1}{6} + 0} = \sqrt{\frac{4+1+1}{6}} = \sqrt{\frac{6}{6}} = \sqrt{1} = 1$.

The following theorem shows that the state preparation for the weighted amplitudes described above can be implemented by a unitary transformation.

Theorem 2. *For any normalized state vector $|v\rangle \in \mathbb{C}^N$, where $N = 2^n$, there exists a unitary matrix $U \in \mathbb{C}^{N \times N}$ such that*

$$U |0\rangle^{\otimes n} = |v\rangle .$$

Here, $|0\rangle^{\otimes n}$ represents the standard initial state $|00...0\rangle$.

Proof. Let $|v\rangle$ be any normalized vector in $\mathbb{C}^N$, satisfying $\langle v|v\rangle = 1$. By the Gram–Schmidt orthonormalization process, $|v\rangle$ can be extended to a full orthonormal basis of $\mathbb{C}^N$. That is, there exist vectors $|v_2\rangle, \ldots, |v_N\rangle$ such that $\{|v\rangle, |v_2\rangle, \ldots, |v_N\rangle\}$ is an orthonormal basis of $\mathbb{C}^N$. We now define the matrix U as $U = \begin{bmatrix} |v\rangle & |v_2\rangle & \cdots & |v_N\rangle \end{bmatrix}$, where each $|v_j\rangle$ is written as a column vector. Since the columns of U form an orthonormal set, it follows that $U^\dagger U = I_N$, where I_N is the $N \times N$ identity matrix. Therefore, U is unitary. Finally, observe that multiplying U by the standard basis vector $|0\rangle = [1, 0, \ldots, 0]^T$ yields $U|0\rangle = |v\rangle$, since $|0\rangle$ selects the first column of U, which is $|v\rangle$ by construction. Thus, such a unitary matrix U exists for any normalized $|v\rangle$, as required. Q.E.D. □

Based on Theorem 2, for any normalized state vector $|v\rangle \in \mathbb{C}^{2^n}$, there exists a unitary matrix U such that $U|0\rangle = |v\rangle$. And a method [27] exists to decompose any unitary matrix U into circuits that require

$$\frac{22}{48}4^n - \frac{3}{2}2^n + \frac{5}{3}$$

CNOT gates. Therefore, any normalized state $|v\rangle$ can be obtained from $|0\rangle$ using a circuit that requires $\frac{22}{48}4^n - \frac{3}{2}2^n + \frac{5}{3}$ CNOT gates.

The circuit described in Section 3 is designed to operate on a uniform superposition (i.e., $H^{\otimes n}|0\rangle^{\otimes n}$). However, its functionality remains valid for the non-uniform superposition prepared here. This is due to the principle of linearity in quantum mechanics: the oracle's logic, implemented with gates like the Toffoli gate, acts on each computational basis state $|v\rangle$ independently. The amplitude α_v associated with each $|v\rangle$ is carried through the transformation unaffected by the logical operations. Therefore, the oracle correctly identifies connected dominating sets regardless of the initial amplitude distribution, allowing our approach to enhance the measurement probability of smaller sets of CDS.

We formalize the circuit from Section 3 as a quantum oracle, denoted by u_f. This oracle acts on an input register $|v\rangle$ and an ancilla qubit $|y\rangle$. It evaluates a function $f(v)$ that determines if the vertex set corresponding to v is a connected dominating set (CDS):

$$f(v) = \begin{cases} 1, & \text{if } v \in \text{CDS} \\ 0, & \text{otherwise.} \end{cases}$$

The action of the oracle is defined as a conditional bit-flip on the ancilla qubit:

$$u_f|v\rangle|y\rangle = |v\rangle|y \oplus f(v)\rangle,$$

where $\oplus$ denotes addition modulo 2. The ancilla qubit is initialized to the state $|-\rangle$:

$$|y\rangle_{\text{initial}} = H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

Then we have

$$u_f|v\rangle|y\rangle = \begin{cases} -1|v\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], & \text{if } v \in \text{CDS}, \\ +1|v\rangle\left[\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right], & \text{otherwise.} \end{cases}$$

After the oracle u_f has marked the solution states by inverting their phase, the next step is to amplify their amplitudes. This is achieved using a weighted reflection operation $g = 2|v\rangle\langle v| - I_N$. By repeatedly applying gu_f to the initial state $|v\rangle$, the amplitudes of the solution states are amplified, allowing a desired CDS to be obtained with high probability upon measurement.

4.1. Analysis of the Success Probability with Weighted Initial State

Standard derivations of Grover's algorithm, such as in [6,7,28], assume a uniform superposition as the initial state: $|v\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$. These standard treatments do not address the case of a weighted initial state, such as the one defined by the amplitude distribution $\alpha_x = \frac{w(x)}{\|\mathbf{w}\|}$ used in this work. Therefore, to ensure the integrity of our proposed method, it is necessary to provide a proof of its correctness.

Recall that our initial state is: $|v\rangle = \sum_{x=0}^{N-1}\alpha_x|x\rangle$. Let $|x_t\rangle$ be the target state and $\frac{\theta}{2}$ be the angle between $|v\rangle$ and horizontal axis, assuming $|x_t\rangle$ is taken as the vertical axis. We first count the angle $\frac{\pi}{2} - \frac{\theta}{2}$ between $|x_t\rangle$ and $|v\rangle$. The inner product $\langle x_t|v\rangle = \alpha_{x_t} = \frac{n-s_2(x_t)}{\sqrt{\sum_{x=0}^{N-1}(n-s_2(x))^2}} = \frac{n-s_2(x_t)}{\sqrt{2^{n-2}n(n+1)}}$. Since $\langle x_t|v\rangle = \|x_t\|\|v\|\cos(\frac{\pi}{2} - \frac{\theta}{2}) =$

$\cos(\frac{\pi}{2}-\frac{\theta}{2}) = \alpha_{x_t}$, when 2^n is large, $\alpha_{x_t} = \cos(\frac{\pi}{2}-\frac{\theta}{2}) \approx 0$ implies $\frac{\pi}{2}-\frac{\theta}{2} \approx \frac{\pi}{2}$. See Figure 21 for visualizing the angle $\frac{\theta}{2}$. Since $\frac{\pi}{2}-\frac{\theta}{2} \approx \frac{\pi}{2}$, the angle $\frac{\theta}{2}$ between $|v\rangle$ and horizontal axis is small. Thus, $\frac{\theta}{2} \approx \sin(\frac{\theta}{2}) = \cos(\frac{\pi}{2}-\frac{\theta}{2})$. Since $\alpha_{x_t} = \langle x_t|v\rangle = \| |x_t\rangle \| \| |v\rangle \| \cos(\frac{\pi}{2}-\frac{\theta}{2}) = \sin(\frac{\theta}{2}) \approx \frac{\theta}{2}$, we have $\theta \approx 2\alpha_{x_t}$. Visualized geometrically, an increase in α_{x_t} corresponds to an increase in θ, indicating that initial states with higher amplitudes are positioned in closer proximity to the target state $|x_t\rangle$.

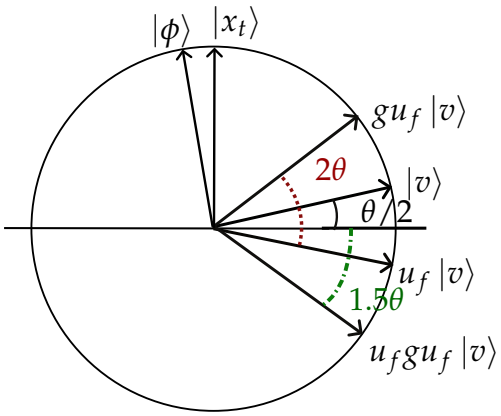


Figure 21. Geometric visualization of the weighted search algorithm in the two-dimensional subspace. The initial state $|v\rangle$ makes angle $\frac{\theta}{2}$ with the horizontal axis, while $|x_t\rangle$ represents the target state (vertical). Each gu_f iteration rotates the quantum state by θ, where $\theta \approx 2\alpha_{x_t}$ depends on the initial target amplitude. After k iterations, the initial state $|v\rangle$ is rotated into a final state, denoted $|\phi\rangle$, that is closely aligned with the target state $|x_t\rangle$.

Oracle u_f flips the sign of the amplitude corresponding to $|x_t\rangle$. That is $u_f|v\rangle = (\sum_{x\neq x_t} \alpha_x |x\rangle) - \alpha_{x_t}|x_t\rangle$ — applying u_f to $|v\rangle$ results in the reflection of $|v\rangle$ across the horizontal axis. The position of $u_f|v\rangle$ is illustrated in Figure 21.

Next, we describe the action of the operator g. Let $u_f|v\rangle = |\psi\rangle$. To understand the action of g on $|\psi\rangle$, we first decompose $|\psi\rangle$ into $|\psi\rangle = a|v\rangle + b|v^\perp\rangle$ — the state $|\psi\rangle$ can be expressed as a linear combination of the mutually perpendicular states $|v\rangle$ and $|v^\perp\rangle$. Then, the operator g is applied to $|\psi\rangle$ producing $g|\psi\rangle = a|v\rangle - b|v^\perp\rangle$ because $g|v\rangle = |v\rangle$ and $g|v^\perp\rangle = (2|v\rangle\langle v| - I_N)|v^\perp\rangle = 2|v\rangle\langle v||v\rangle^\perp - |v^\perp\rangle = -|v^\perp\rangle$ where $\langle v|v^\perp\rangle = 0$. This result shows that the component of $|\psi\rangle$ parallel to $|v\rangle$ is unchanged, while the component orthogonal to $|v\rangle$ is inverted. This is precisely the definition of a reflection of the state $|\psi\rangle$ across the axis defined by $|v\rangle$. The position of $gu_f|v\rangle$ is illustrated in Figure 21.

To summarize the analysis presented above, the application of gu_f to $|v\rangle$ results in a rotation of $|v\rangle$ toward $|x_t\rangle$ by angle θ, with the rotation angle θ increasing as α_{x_t} increases. Now, we proceed to determine the required number of iterations of the gu_f operator and analyze the probability for finding our target state $|x_t\rangle$.

Since the initial angle between $|v\rangle$ and the horizontal axis is $\frac{\theta}{2}$, each application of the operator gu_f rotates the state toward $|x_t\rangle$ by an additional angle of θ. We expect that after performing k iterations of gu_f, the resulting state $|\phi\rangle$ forms a very small angle with $|x_t\rangle$. Thus, we have $k\theta = \frac{\pi}{2}-\frac{\theta}{2}$, which gives $k = \left\lfloor \frac{\pi}{2\theta} - \frac{1}{2} \right\rfloor \approx \left\lfloor \frac{\pi}{4\alpha_{x_t}} - \frac{1}{2} \right\rfloor$. After k iterations, the state of the qubits is $|\phi\rangle \approx \left(gu_f\right)^{\lfloor \frac{\pi}{4\alpha_{x_t}} - \frac{1}{2} \rfloor} |v\rangle$. The maximum angle between $|\phi\rangle$ and $|x_t\rangle$ is $\frac{\theta}{2}$. Let ℓ be the length of the projection of $|\phi\rangle$ onto $|x_t\rangle$. Thus, when we measure the state $|\phi\rangle$, the probability of obtaining $|x_t\rangle$ is $|\ell|^2$, where $\ell = \langle x_t|\phi\rangle$ is the projection amplitude of $|\phi\rangle$ onto $|x_t\rangle$. Let ϵ be the small angle between the final state $|\phi\rangle$ and the target $|x_t\rangle$. From the relation $\ell = \langle x_t|\phi\rangle = \||x_t\rangle\| \||\phi\rangle\| \cos(\epsilon)$, and under the normalization $\||x_t\rangle\| = \||\phi\rangle\| = 1$, it follows that $\ell = \cos(\epsilon)$. Since this angle is bounded by $\epsilon \leq \theta/2$, the success probability is lower bounded by: $|\ell|^2 = \cos^2(\epsilon) \geq \cos^2(\theta/2) = 1 - \sin^2(\theta/2) \approx 1 - \alpha_{x_t}^2$, which is very close to 1 since α_{x_t} is small.

In the Grover's search, if M (the number of solutions) is unknown, the optimal iteration count k cannot be computed exactly. To handle this, we use an adaptive strategy. The algorithm runs step by step with k adjusted based on measurement results. We begin with an initial guess $k_0 \approx \frac{\pi}{4}\sqrt{2^n}$. The optimal number of iterations for Grover's algorithm is approximately $(\pi/4)\sqrt{2^n/M}$, where M is the number of solutions. Since M is unknown a priori, we adopt the most difficult case where there is only a single solution ($M = 1$). After each measurement, we check the

output bitstring x with the oracle (from Section 3) to see if it is a valid CDS. Counting the successes over multiple trials gives $\hat{P}$. If $\hat{P}$ is too low, we increase k; if $\hat{P}$ starts to decrease, we reduce k. This feedback allows k to adaptively converge to a near-optimal value.

Grover operators u_f and g can be represented as 2^n-dimensional unitary matrices. Based on the previously mentioned method [27], each of them can be implemented using $\frac{22}{48}4^n - \frac{3}{2}2^n + \frac{5}{3}$ CNOT gates. Since the operator gu_f is repeated $k \approx \left\lfloor \frac{\pi}{4\alpha_{x_t}} - \frac{1}{2} \right\rfloor$ times, the total number of CNOT gates required by the proposed non-uniform Grover search for the minimum CDS is $\left(2\left\lfloor \frac{\pi}{4\alpha_{x_t}} - \frac{1}{2} \right\rfloor + 1\right)\left(\frac{22}{48}4^n - \frac{3}{2}2^n + \frac{5}{3}\right)$ CNOT gates.

4.2. Analysis of Measurement Probability Differences between CDSs of Different Sizes

This section provides an analysis to estimate the difference in the final measurement probability between two solution states whose number of 0 differ by one. This analysis relies on the geometric intuition of the amplitude amplification process derived previously.

A complete mathematical derivation of the probability difference ΔP is hard because we do not know all the required information in advance. To obtain an exact formula such as $\Delta P = f(n, M, w(x_1), w(x_2), \dots)$, we would need to know how many solutions there are (M) and what the weight of each solution is. Since this information is unavailable, we focus on analyzing one target state at a time. Using simple approximations (for example, $\theta \approx 2\alpha_{x_t}$), we can still understand how the final measurement probability depends on the initial amplitude of a solution.

Let $|x_1\rangle$ and $|x_2\rangle$ be two distinct target states (Connected Dominating Sets) corresponding to bitstrings whose number of 1s differ by one:

$$s_2(x_2) = s_2(x_1) + 1,$$

where, $s_2(x)$ denotes the number of ones in the bitstring x. By the definition of the weight function $w(x) = n - s_2(x)$:

$$\begin{aligned} w(x_1) &= n - s_2(x_1) \\ w(x_2) &= n - s_2(x_2) = n - (s_2(x_1) + 1) = w(x_1) - 1. \end{aligned}$$

The state $|x_1\rangle$ is the preferred solution as it corresponds to a smaller CDS (fewer ones, more zeros). The initial amplitudes are:

$$\alpha_{x_1} = \frac{w(x_1)}{\|\mathbf{w}\|} \quad \text{and} \quad \alpha_{x_2} = \frac{w(x_1) - 1}{\|\mathbf{w}\|}.$$

This confirms that the preferred solution $|x_1\rangle$ has a greater initial amplitude: $\alpha_{x_1} > \alpha_{x_2}$.

We adopt the approximations derived from the weighted amplitude amplification (where the angle of rotation θ is approximately $2\alpha_{x_t}$).

The number of iterations required to optimally amplify the preferred state $|x_1\rangle$ is:

$$k_1 \approx \left\lfloor \frac{\pi}{4\alpha_{x_1}} - \frac{1}{2} \right\rfloor.$$

We run the algorithm for this fixed number of iterations, resulting in the final state $|\phi\rangle$.

The final probability of measuring $|x_1\rangle$, $P(x_1)$, is given by $P(x_1) = \cos^2(\epsilon_1)$, where ϵ_1 is the small angle between $|\phi\rangle$ and $|x_1\rangle$. When number of iterations is k_1, ϵ_1 is bounded by $\frac{\theta_1}{2}$:

$$P(x_1) \approx \cos^2\left(\frac{\theta_1}{2}\right) = 1 - \sin^2\left(\frac{\theta_1}{2}\right) \approx 1 - \alpha_{x_1}^2.$$

Since α_{x_1} is small for large n, this probability $P(x_1)$ is very close to 1.

Each Grover iteration, gu_f, rotates the state vector by an angle of θ_2 towards $|x_2\rangle$. The algorithm is executed for k_1 iterations, where k_1 is the number of iterations optimized for the preferred target $|x_1\rangle$, not for $|x_2\rangle$. After k_1 iterations, the total angle θ_{total} of the state vector with respect to the horizontal axis is given by the initial angle plus the accumulated rotation:

$$\theta_{\text{total}} = \frac{\theta_2}{2} + k_1\theta_2.$$

The quantity of interest is the final angle, ϵ_2, between the final state $|\phi\rangle$ and the target axis $|x_2\rangle$ (which is at angle $\pi/2$). This angle is the absolute difference between their final angular positions:

$$\epsilon_2 = \left| \left(\frac{\theta_2}{2} + k_1\theta_2 \right) - \frac{\pi}{2} \right|.$$

From the analysis for the preferred state $|x_1\rangle$, we know that the optimal number of iterations is approximately:

$$k_1 \approx \frac{\pi}{2\theta_1} - \frac{1}{2}.$$

We substitute this expression for k_1 into the equation for ϵ_2:

$$\epsilon_2 \approx \left| \left(\frac{\theta_2}{2} + \left(\frac{\pi}{2\theta_1} - \frac{1}{2} \right)\theta_2 \right) - \frac{\pi}{2} \right|.$$

We now simplify the expression inside the absolute value. First, distribute θ_2:

$$\epsilon_2 \approx \left| \frac{\theta_2}{2} + \frac{\pi\theta_2}{2\theta_1} - \frac{\theta_2}{2} - \frac{\pi}{2} \right| = \left| \frac{\pi\theta_2}{2\theta_1} - \frac{\pi}{2} \right| = \frac{\pi}{2} \left| 1 - \frac{\theta_2}{\theta_1} \right|.$$

Finally, using the approximation $\theta \approx 2\alpha$, we can express the angle in terms of the initial amplitudes:

$$\epsilon_2 \approx \frac{\pi}{2} \left| 1 - \frac{2\alpha_{x_2}}{2\alpha_{x_1}} \right| = \frac{\pi}{2} \left| 1 - \frac{\alpha_{x_2}}{\alpha_{x_1}} \right|.$$

We analyze the ratio $\alpha_{x_2}/\alpha_{x_1}$ for the two target states:

$$\frac{\alpha_{x_2}}{\alpha_{x_1}} = \frac{w(x_1) - 1}{w(x_1)} = 1 - \frac{1}{w(x_1)}.$$

Substituting this ratio back into the expression for the final angle ϵ_2:

$$\epsilon_2 \approx \frac{\pi}{2} \left| 1 - \left(1 - \frac{1}{w(x_1)} \right) \right| = \frac{\pi}{2w(x_1)}.$$

Since $w(x_1)$ is large (it is at most n), ϵ_2 is a small angle. We utilize the small-angle approximation for the cosine function: $\cos(\delta) \approx 1 - \frac{\delta^2}{2}$ (based on Taylor Series for $\cos(\delta)$), which gives $\cos^2(\delta) = 1^2 - 2\frac{\delta^2}{2} + \frac{\delta^4}{4} \approx 1 - \delta^2$. The final probability of measuring $|x_2\rangle$ is:

$$P(x_2) \approx 1 - \epsilon_2^2 \approx 1 - \left(\frac{\pi}{2w(x_1)} \right)^2 = 1 - \frac{\pi^2}{4w(x_1)^2}.$$

The difference in the final measurement probabilities ΔP is:

$$\begin{aligned} \Delta P &= P(x_1) - P(x_2) \\ &\approx (1 - \alpha_{x_1}^2) - \left(1 - \frac{\pi^2}{4w(x_1)^2} \right) \\ &= \frac{\pi^2}{4w(x_1)^2} - \alpha_{x_1}^2. \end{aligned} \tag{4}$$

To confirm that $\Delta P > 0$, we must show that the first term in Equation 4 is greater than the second term, i.e., $\frac{\pi^2}{4w(x_1)^2} > \alpha_{x_1}^2$. By substituting the definition $\alpha_{x_1} = w(x_1)/\|\mathbf{w}\|$, this inequality can be rewritten as:

$$\frac{\pi^2}{4w(x_1)^2} > \frac{w(x_1)^2}{\|\mathbf{w}\|^2},$$

which is equivalent to showing that $\pi^2\|\mathbf{w}\|^2 > 4w(x_1)^4$. Using the known normalization factor described earlier, $\|\mathbf{w}\|^2 = 2^{n-2}n(n+1)$, our condition becomes:

$$\pi^2 \cdot 2^{n-2}n(n+1) > 4(n - s_2(x_1))^4.$$

The left-hand side of this inequality grows exponentially with the number of qubits n (dominated by the 2^n term), while the right-hand side grows only polynomially ($O(n^4)$). This condition therefore holds for any reasonably large n, which rigorously confirms that $\Delta P > 0$ and thus $P(x_1) > P(x_2)$.

Using mathematical induction, one can prove that the inequality remains valid for all $n \geq 6$ under the worst-case condition $s_2(x_1) = 0$. Since practical graph problems typically involve n well above this threshold, we can confidently conclude that $\Delta P > 0$ for any meaningful problem size.

By selecting the iteration count k_1 optimized for the preferred solution $|x_1\rangle$, the algorithm leverages this initial advantage, resulting in a final probability distribution that significantly favors the smaller CDS $|x_1\rangle$ over the larger one $|x_2\rangle$.

We summarize the results of this section with the following theorem.

Theorem 3. The minimum connected dominating set (CDS) can be found in $O(\sqrt{2^n})$ time using quantum computing. The algorithm, which prepares a weighted initial state prioritizing smaller CDSs via amplitudes $\alpha_x = w(x)/\|\mathbf{w}\|$ (with $w(x) = n - s_2(x)$), ensures that the measurement probability of a smaller CDS exceeds that of a larger one for graphs with $n \geq 6$ vertices.

5. The Implementation and Simulation Results

5.1. The Implementation

To demonstrate and verify the proposed scheme, we use `Qiskit` [29], an open-source software framework for quantum computing, to implement the algorithm and test it using the problem instance described in the previous sections.

The problem instance is the graph $G = (V, E)$ where $V = \{v_5, v_4, v_3, v_2, v_1, v_0\}$ and $E = \{e_{01}, e_{12}, e_{14}, e_{23}, e_{34}, e_{45}\}$. In Figure 10, the visual representation of this graph is shown. To demonstrate the impact of graph complexity on solving time, we also tested our proposed method using another graph, G'. Graph G' is formed by removing edge e_{14} from graph G. That is, $G' = (V' = V, E' = E - \{e_{14}\})$. We used `Qiskit` to implement the method proposed in this paper to find the smallest connected dominating set on G and G'. The library `Qiskit` has been adopted as a standard tool by many quantum computers, so our developed programs can be executed on real quantum computers. Given that access to real quantum computers is limited, this paper uses `Qiskit`'s simulation component, `Qiskit Aer`, to conduct our experiments. The simulation method used in this paper is `matrix_product_state`, one of several simulation methods available in `Qiskit Aer`. The simulation method `matrix_product_state`, originally proposed by Vidal [30], is further detailed in [31]. All simulation experiments were conducted on a computer running the Ubuntu 22.04 operating system and using an AMD Ryzen 7 3700X 8-core processor. This CPU has a clock speed of 2200 MHz and a cache size of 512 KB.

Next, we describe how to implement the proposed method for finding the connected dominating set of graph G. The circuit for finding the dominating set on G' is similar to that of G, except for the removal of the components related to edge e_{14}. Six qubits $|\mathbf{v}\rangle = |v_5v_4v_3v_2v_1v_0\rangle$ are used to represent vertices, and six qubits $|e_{01}\rangle, |e_{12}\rangle, |e_{14}\rangle, |e_{23}\rangle, |e_{34}\rangle, |e_{45}\rangle$ are used to represent edges. Qubits $|e_{01}e_{12}e_{14}e_{23}e_{34}e_{45}\rangle$ are initialized to $|111111\rangle$. In the initial stage, v_i are initialized to $|0\rangle$ where $0 \leq i \leq 5$ and Hadamard gates are used to transform each of the six vertices into a superposition state, allowing the simultaneous execution of the dominating test and the connecting test on all vertex subsets. That is $H|v_i\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ where $0 \leq i \leq 5$. Therefore, after completing all the processing stages and measuring v_i, there is a $\frac{1}{2}$ chance of obtaining a measurement value of 0 (indicating that this vertex is not included in the candidate set for the dominating set) and a $\frac{1}{2}$ chance of obtaining a measurement value of 1 (indicating that this vertex is included in the candidate set for the dominating set). After preparing the superposition of vertices, six additional auxiliary qubits $|\hat{\mathbf{v}}\rangle = |\hat{v}_5\hat{v}_4\hat{v}_3\hat{v}_2\hat{v}_1\hat{v}_0\rangle$ are used to implement the dominating test process.

Now, we are ready to implement the dominating test process. The circuit shown in Figure 22 implements the method mentioned in Section 3.2. For simplicity, we only show $QC2_{01}$ and $QC2_{10}$ in Figure 22. Using the same construction rules as in Figure 22, readers can extend this to obtain $QC2_{12}$, $QC2_{21}$, $QC2_{23}$, $QC2_{32}$, $QC2_{34}$, $QC2_{43}$, $QC2_{45}$, $QC2_{54}$, $QC2_{14}$, and $QC2_{41}$. Notice in Figure 22 that we use aux_0 and aux_1 to respectively store $|\overline{v_i}\rangle$ and $|\overline{\hat{v}_i}\rangle$

where $0 \leq i \leq 5$. Also, aux_0 and aux_1 can be reused in QC2_{12}, QC2_{21}, QC2_{23}, QC2_{32}, QC2_{34}, QC2_{43}, QC2_{45}, QC2_{54}, QC2_{14}, and QC2_{41}. After completing QC2_{01}, QC2_{10}, QC2_{12}, QC2_{21}, QC2_{23}, QC2_{32}, QC2_{34}, QC2_{43}, QC2_{45}, QC2_{54}, QC2_{14}, and QC2_{41}, the circuit in Figure 11(b) is employed to detect whether $|v\rangle_i$ where $0 \leq i \leq 5$ are all set to 1. If they are all set to 1, $|\varphi_{\text{dominated}}\rangle$ is set to 1; otherwise, $|\varphi_{\text{dominated}}\rangle$ is set to 0.

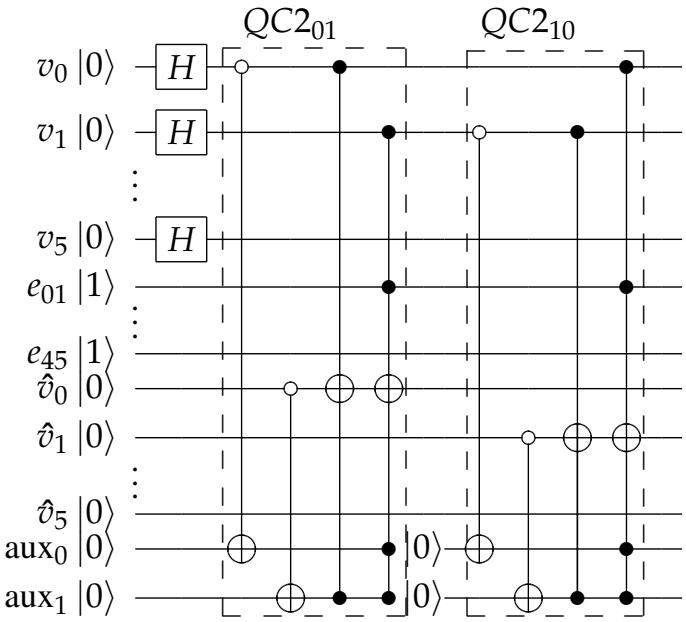


Figure 22. QC2_{01} and QC2_{10}.

After completing the implementation of the dominating test, we then implemented the connecting test. First, the circuit uses a controlled NOT gate to implement the setting of the value of $\hat{v}_k$ based on the quality of v_k as described in Algorithm `Process Connected-test`, line 4. See the leftmost CNOT in Figure 23 for illustration. Then $QC3_{01}$, $QC3_{10}$,$QC3_{12}$, $QC3_{21}$, $QC3_{23}$, $QC3_{32}$,$QC3_{34}$, $QC3_{43}$, $QC3_{45}$, $QC3_{54}$,$QC3_{14}$, and $QC3_{41}$ are implemented as shown in Figure 23. Note the repeated structure as illustrated in Figure 13. A similar circuit to that shown in Figure 23 will be repeated six times in total, with a different seed $\hat{v}_k$ set at the beginning of each iteration. Finally, the circuit shown in Figure 17 was implemented to integrate the six sub-circuits to determine $|\phi_{\text{connected}}\rangle$.

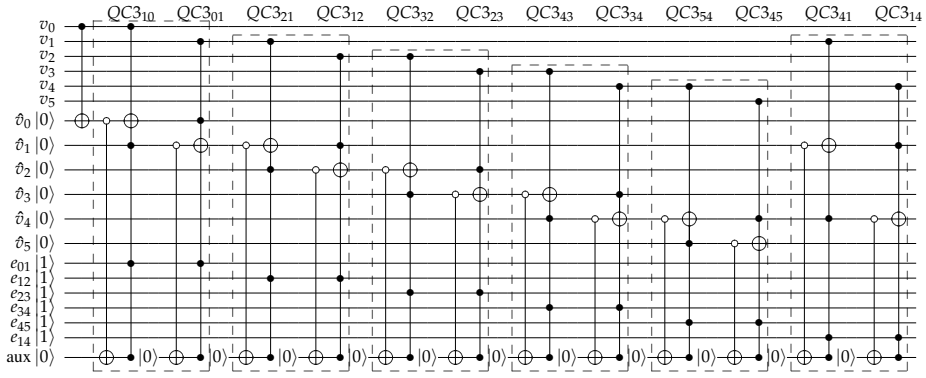


Figure 23. Implementation of $QC3_{01}$, $QC3_{10}$, $QC3_{12}$, $QC3_{21}$, $QC3_{23}$, $QC3_{32}$, $QC3_{34}$, $QC3_{43}$, $QC3_{45}$, $QC3_{54}$, $QC3_{14}$, and $QC3_{41}$.

After completing the connecting test, we implemented two methods to determine the size of the dominating set: one using half-adders and the other using truth tables.

In the truth table approach, we use Toffoli gates to implement the circuit for converting unary to binary representation. See Figure 24 for an illustration of the circuit used to calculate the size of the dominating set. Here, we

reuse $|\hat{v}_2\hat{v}_1\hat{v}_0\rangle$ to store the binary representation of the dominating set size. For example, if there are five '1's in $|v_5v_4v_3v_2v_1v_0\rangle = |110111\rangle$, then $|\hat{v}_2\hat{v}_1\hat{v}_0\rangle = |101\rangle$.

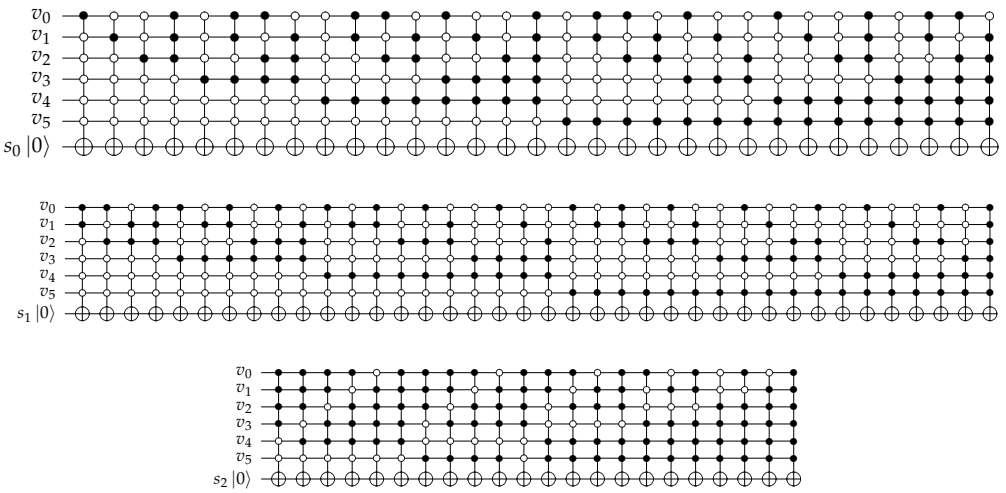


Figure 24. Implementation of the truth table approach for finding the size of the dominating set.

In Figure 25, a circuit utilizing several half adders to calculate the size of the dominating set and convert it to a binary representation is implemented. Notice that the maximum dominating set size in this example is 6, which is not exactly a power of 2. As a result, some intermediate results calculated by the half adders will not be used in determining the final result. For example, t_{16}, t_{19}, and t_{20} can be ignored in this instance. Although t_{16}, t_{19}, and t_{20} in Figure 25 can be disregarded and some auxiliary qubits can be reused, for the sake of clarity in demonstrating the relationships between the half adders, we present the complete input and output relationships of all the half adders.

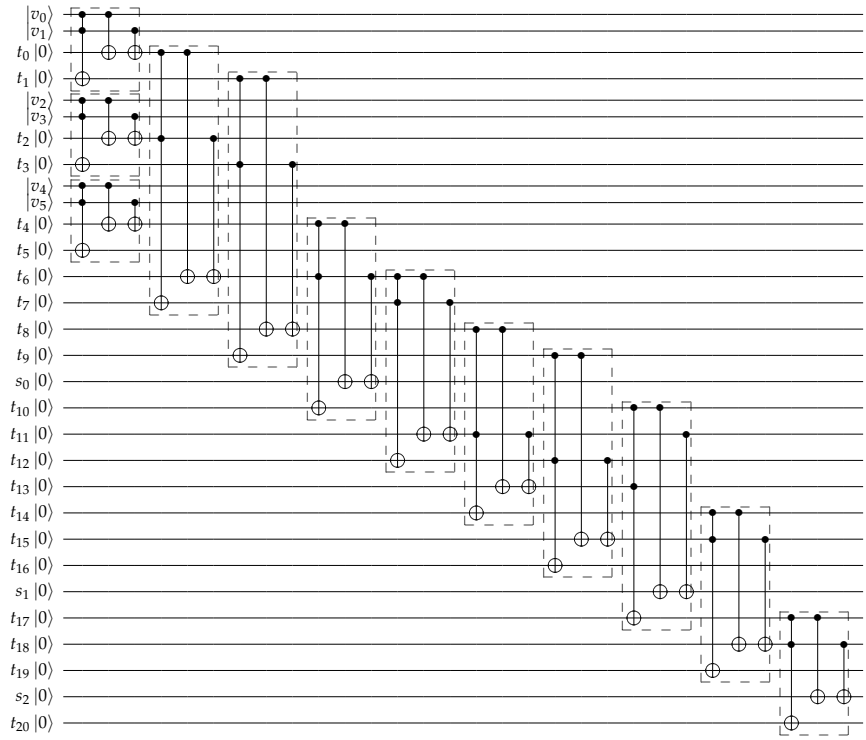


Figure 25. The implementation of the half-adder approach for finding the size of the dominating set.

In Figure 26, we compare the difference in computation time caused by using the truth table approach (the fast adder approach) versus the half adder approach, while keeping all other circuits identical. The x-axis represents the

number of measurements, and the y-axis represents the computation time required. From the figure, we can observe that as the number of shots increases, the required time increases linearly. Additionally, the time required using the half adder approach remains consistently higher compared to the truth table approach.

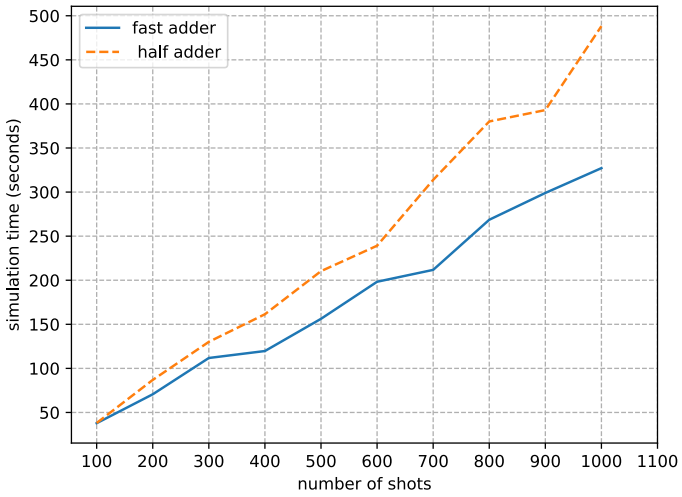


Figure 26. Comparison of the simulation time of graph G between using fast adder and half adder. The x-axis is the number of shots in the simulation, and the y-axis represents the simulation time (in seconds).

To demonstrate the impact of different edge counts in the graph, we removed edge e_{14} from graph G to create graph G'. It is evident that the minimum connected dominating set of G is $\{v_1, v_4\}$, while the minimum connected dominating set of G' is $\{v_1, v_2, v_3, v_4\}$. The aforementioned experiments conducted on graph G were also applied to graph G', and the results are shown in Figure 27. We compared the total computation time required to calculate the connected dominating set and size of the connected dominating set of graph G' using both the half adder approach and the truth table approach under different numbers of shots. Based on Figure 27, we observe a trend similar to that in Figure 26: as the number of shots increases, the computation time also increases linearly. Additionally, the time required using the truth table approach remains consistently lower than that of the half adder approach. By comparing Figure 26 with Figure 27, we observe that, for the same number of nodes, a lower number of edges results in reduced computational time. This result aligns with theoretical expectations. Although the number of nodes remains unchanged, an increase in the number of edges leads to an increase in the number of sub-circuits in $QC2$, $QC3$, and $QC4$, thereby extending the overall computation path length.

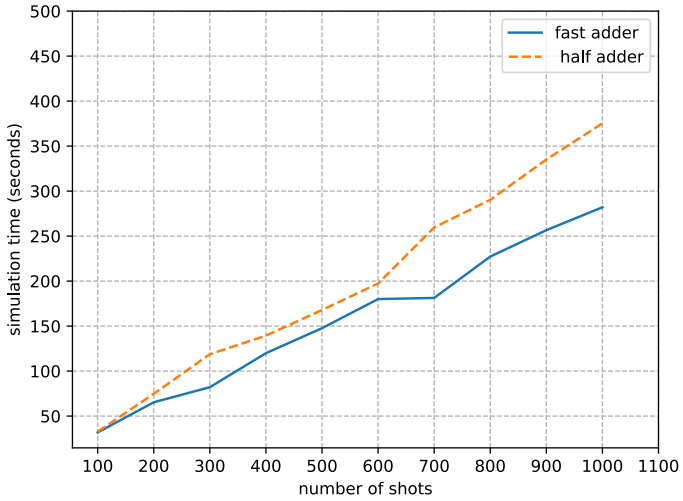


Figure 27. Comparison of the simulation time of graph $G' = (V, E - \{e_{14}\})$ between using fast adder and half adder The x-axis represents the number of shots in the simulation, and the y-axis represents the simulation time (in seconds).

5.2. Analysis of Qubit and Universal Quantum Gate Usage

Now, we try to analyze the worst case bound for the number of qubits and the universal quantum gates used in our proposed method. Note that the proposed method consists of 5 processes, namely, the *state enumeration*, the *dominating test process*, the *connected test process*, the *size computation process*, and the *results output & measurement*. In the following, we will examine the number of qubits used in each process. Note that, since the process of the *results output & measurement* takes no auxiliary qubits used, we omit it here.

5.2.1. State Enumeration

In this process, the problem instance input takes n qubits for $|\varphi_{vertex}>$ and m qubits for $|\varphi_{edge}>$, where n (and m) is equal to $|V|$ (and $|E|$). The total number of qubits used in this process is equal to $n+m$. And it takes n hadamard gates to generate the solution space in this process.

5.2.2. Dominating Test Process

In this process, the additional auxiliary qubits used in the quantum circuit $QC2\_a$ (see Figure 10(b) and Figure 11(a)) are the dominated flag qubit set $|\varphi_{\hat{vertex}}>$ plus 2 temporarily auxiliary qubits. It is because the auxiliary qubits used for quantum circuit $QC2_{ij}$ are equal to 2, for each $e_{ij} \in E$, and each $QC2_{ij}$ in the quantum circuit of $QC2\_a$ is performed sequentially. Thus, the auxiliary qubits can be reused. Thus, the number of qubits used for the quantum circuit $QC2\_a$ is equal to $n+2$. The number of qubits used for the quantum circuit $QC2\_b$ (see Figure 11(b)) is equal to

$$\begin{aligned} & \frac{n}{4^1} + \frac{n}{4^2} + \ldots + \frac{n}{4^{\log_4 n}} \\ = \; & n \cdot [\frac{1}{4^1} + \frac{1}{4^2} + \ldots \frac{1}{4^{log_4 n}}] \\ = \; & \frac{n}{3} \cdot [1 - (\frac{1}{4})^{log_4 n}] \\ = \; & \frac{n}{3} \cdot [1 - \frac{1}{n}] \\ = \; & \frac{n-1}{3} \end{aligned} \tag{5}$$

In addition, the result will be stored in a characteristic determination qubit $|\varphi_{dominated}>$, which takes 1 qubit. Summarizing the above discussions, we conclude that the total number of additional auxiliary qubits used for the *dominating test process* is equal to $n+2+\lceil(n-1)/3\rceil+1 = n+\lceil(n-1)/3\rceil+3$.

For the analysis of universal quantum gate usage, each Toffoli gate (or CCCCNOT gate) can be implemented using 2 (or 4) CNOT gates. Accordingly, the quantum circuit $QC2_{ij}$ (see Figure 10(b)) requires 8 universal quantum gates. Since the quantum circuit $QC2_a$ contains a loop that iterates m times, and each iteration executes both $QC2_{ij}$ and $QC2_{ji}$, the total number of universal quantum gates required for $QC2_a$ is $16m$. The quantum circuit QC_b shown in Figure 11(b), use 4 auxiliary qubits per group and applies the CCCCNOT gate to generate intermediate results. This process is then applied recursively in a tree-like manner. The total number of universal quantum gates required to implement $QC2_b$ can thus be calculated as: $4\cdot\frac{n}{4^1}+4\cdot\frac{n}{4^2}+\ldots+4\cdot\frac{n}{4^{\log_4 n}} = \frac{4}{3}(n-1)$. Therefore, the total number of universal quantum gates required to implement the dominating test process, which comprises both $QC2_a$ and $QC2_b$, is equal to $16m+\frac{4}{3}(n-1)$.

Connected Test Process

In this process, the additional auxiliary qubits are the connected flag qubit set $|\varphi_{\hat{vertex}}>$, which can reused the auxiliary qubits set in the *dominating test process*; thus, the additional number of qubits count is 0. For the quantum circuit $QC3^k_{\text{Reach-a}}$ and $QC3^k_{\text{Reach-b}}$ (see Figures 12-14), since the quantum operations can be performed sequentially, the auxiliary qubits can reuse the previous qubits; thus, the additional number of qubits count is also 0. Concluding the above discussions, we have that the total number of additional auxiliary qubits used for this process is only one characteristic determination qubit, the $|\varphi_{connected}>$.

For the analysis of universal quantum gate usage in the reachability testing phase, the quantum circuit $QC3_{ij}$ (see Figure 12(b)) requires 5 universal quantum gates. The quantum circuit $QC3^k_{Reach-a}$ involves a triple nested loop, where each iteration executes both $QC3_{ij}$ and $QC3_{ji}$. Therefore, the total number of universal quantum gates required for $QC3^k_{Reach-a}$ is $10mn^2$. Additionally, the quantum circuit $QC3^k_{Reach-b}$ performs n executions of $QC3_i$, each requiring 5 universal quantum gates (see Figure 14(a)). Hence, the total number of gates required for $QC3^k_{Reach-b}$ is $5n$. Combining the above, the total number of universal quantum gates required to implement the reachability testing phase is $10mn^2+5n$. Finally, the overall connected test quantum circuit $QC3$ consists of n executions of $QC3^i_{Result-integrate}$, each of which requires $10mn^2+5n+6$ universal quantum gates. Therefore, the total number of gates required for the complete connected test process is $n\cdot(10mn^2+5n+6) = 10mn^3+5n^2+6n$.

Size Computation Process

In this process, the auxiliary qubits to store the result are $|\varphi_{size}>$, which takes $\lceil\log_2 n\rceil+1$ additional qubits. For the Half adder approach (see Figure 18), the additional qubits used are equal to the number of carry bits and sum bits in the first column half adder of the quantum circuit (the other half adders can reuse the auxiliary qubits). Thus, there is a total of n qubits to play the roles of carry bits and sum bits. The total number of additional qubits used in this process (using the Half adder approach) is equal to $n+\lceil\log_2 n\rceil+1$. Similarly, the same arguments can be made on the Toffoli gate approach, and the total number of qubits used in this process is equal to $\lceil\log_2 n\rceil+1$.

For the analysis of universal quantum gate usage in the size computation process, the Toffoli gate approach uses a truth table of size 2^n rows to determine each of the size-resulting auxiliary qubits s_0, $s_1,\ldots,s_{\lceil\log_2 n\rceil}$. For each resulting qubit s_i, where $1\le i\le\lceil\log_2 n\rceil$, the implementation depends on the number of $0's$ (or $1's$) in the corresponding result column (see Figure 19(b)). In the worst case, this requires 2^{n-1} Toffoli gates. Since each Toffoli like gate requires n universal quantum gates to implement, the total number of universal quantum gates required in the worst case is $n\cdot\log_2 n\cdot 2^{n-1}$. For the alternative Half adder approach, the gates complexity is similar to that of the Toffoli gate approach, and thus it is omitted here for brevity.

Now, summarizing the above discussions and adding the number of qubits used in each process, for the worst case we will have $(n+m)+(n+\lceil(n-1)/3\rceil+3)+1+(n+\lceil\log_2 n\rceil+1) = m+3n+\lceil(n-1)/3\rceil+\lceil\log_2 n\rceil+5$. The total number of universal quantum gates required for the complete process is given by the sum of $n+(16m+\frac{4}{3}(n-1))+(10mn^3+5n^2+6n)+(n\cdot\ log_2 n\cdot 2^{n-1}) = n\cdot\log_2 n\cdot 2^{n-1}+10mn^3+5n^2+16m+\frac{25}{3}n-\frac{4}{3}$. Thus, we have the following lemma.

Lemma 3. *The number of qubits used for performing the proposed method is equal to* $m+3n+\lceil(n-1)/3\rceil+\lceil\log_2 n\rceil+5 = O(n+m)$. *In terms of gate complexity, the proposed methods requires* $O(n\cdot\log_2 n\cdot 2^{n-1})$ *universal quantum gates.*

Before concluding this subsection, we note that quantum computers are still in a very early stage of development compared to conventional computer architectures. As a result, the use of qubits and quantum gates remains a critically scarce resource, making their optimization an essential concern in quantum algorithm design. Since the method proposed in this paper is presented with an emphasis on accessibility for readers, there remains considerable room for improvement in terms of resource efficiency. Below, we propose suggestions for the implementation phase, focusing on principled design decisions. For example, in the dominating test process, during the dominating-test qubit characteristic determination operations (see Figure 11(b)), our original design groups four dominating-test qubits at a time and uses one ancilla qubit to compute a temporary result. These partial results are then combined recursively, layer by layer, until all results are consolidated and written into qubit $|\varphi_{dominated}>$. According to the computation in Equation (5), this approach requires $\frac{(n-1)}{3}$ ancilla qubits. However, by reusing ancilla qubits across layers, only the first-layer ancilla qubits are necessary, that is, just $\frac{(n)}{4}$ ancilla qubits in total. Furthermore, an even more fine-grained ancilla reuse strategy could be adopted to further reduce the total number of ancilla qubits required.

From the perspective of universal quantum gate usage, a similar discussion applies. In the design of the dominating test flag computation, when implementing circuit $QC2_b$, the method can be generalized by grouping k qubits at a time (with the current implementation using $k = 4$, as shown in Figure 11(b)). Under this generalization, the number of universal quantum gates required can be expressed as $k \cdot \frac{n}{k^1} + k \cdot \frac{n}{k^2} + \ldots + k \cdot \frac{n}{k^{\log_k n}} = \frac{k}{k-1}(n-1)$. This result indicates that as the value of k increases, the total number of universal quantum gates required to implement $QC2_b$ decreases accordingly. The above discussions illustrate that by reducing both qubit and quantum gate usage, the proposed design can achieve more efficient execution and resource savings. A similar optimization analysis can also be applied to the design of the connectivity test flags; however, detailed discussion is omitted here for brevity.

6. Conclusion

In this paper, we first propose a quantum algorithmic circuit design framework aimed at easing the learning curve for algorithm designers entering the field of quantum algorithm development. Based on this framework, we then present a quantum-logic algorithm for solving the Connected Dominating Set (CDS) problem as an example. The proposed solution method adopts a structured algorithmic approach to illustrate the sequence of quantum operations. This paper also discusses several quantum-logic algorithm design techniques from a computer science perspective, forming the foundation of the proposed framework. A correctness proof is provided for one of the core design techniques frequently used in our solution. We extended this with a weighted Grover's algorithm using non-uniform amplitudes that bias search toward smaller CDSs, achieving quadratic speedup. Unlike standard approaches, our method naturally prioritizes optimal solutions. To verify the proposed method, we implement it for the CDS problem and validate its correctness using a numerical example on IBMs Qiskit quantum simulator. Furthermore, we analyze the number of qubits required by the proposed method. In future work, we suggest the development of additional practical quantum-logic algorithm design techniques to further simplify the quantum algorithm design process for researchers.

Author Contributions

Chu-Fu Wang: Supervision, Conceptualization, Methodology, Writing-Original draft preparation, Writing-Reviewing and Editing; Yih-Kai Lin: Conceptualization, Methodology, Writing-Original draft preparation, Writing-Reviewing and Editing; Chia-Ho Ou: Writing-Original draft preparation, Writing-Reviewing and Editing, Validation; Jau-Der Shih: Writing-Original draft preparation, Writing-Reviewing and Editing, Validation. All authors have read and agreed to the published version of the manuscript.

Funding

This research was funded by the National Science and Technology Council (NSTC), Taiwan, under Grant No. NSTC 114-2221-E-153-005.

Conflicts of Interest Statement

The authors declare no conflicts of interest.

Data Availability Statement

No new data were generated or analyzed in this study.

Acknowledgments

This research was supported by the National Science and Technology Council (NSTC), Taiwan, under Grant No. NSTC 114-2221-E-153-005.

References

1. P.W. Shor (1994). Algorithms for quantum computation: Discrete log and factoring, in Proceeding of the 35th Annual Symposium on Foundations of Computer Science, pp.124-134.
2. P.W. Shor (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *Society for Industrial and Applied Mathematics Journal on Computing*, Vol.26, No.5, pp.1484-1509.
3. E. Rieffel and W. Polak (2000). An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, Vol.32, Issue 3, pp.300-335.
4. A. Montanaro (2016). Quantum algorithms: an overview. *NPJ Quantum Information*, Vol.2.
5. Quantum logic gate (2024). Wikipedia:Quantum logic gate Wikimedia Foundation. en.wikipedia.org/wiki/Quantum-logic-gate.
6. L.K. Grover (1996). A fast quantum mechanical algorithm for database search, In the Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, pp.212-219.
7. L.K. Grover (1998). A framework for fast quantum mechanical algorithms, In the Proceedings of the 30th Annual ACM Symposium on the Theory of Computing, pp.53-62.
8. S. Pabst and Y. Nam (2022). A quantum algorithm for network reliability. *Quantum Physics*.
9. Y.J. Zhang, X.D. Mua, X.W. Liu, X.Y. Wang, X. Zhang, K. Li, T.Y. Wub, D. Zhao and C. Dong (2022). Applying the quantum approximate optimization algorithm to the minimum vertex cover problem. *Applied Soft Computing*, Vol.118, pp.1-8.
10. R. Wong, W.L. Chang, W.Y. Chung and A.V. Vasilakos (2023). Biomolecular and quantum algorithms for the dominating set problem in arbitrary networks. *Scientific Reports*, Vol.13, Article No.4205.
11. C.H. Bennett, P. Bernstein, P. Brassard and P. Vazirani (1997). Strengths and weaknesses of quantum computing, *SIAM Journal on Computing*, Vol.26, Issue 5, pp1510-1523.
12. F. Glover, G. Kochenberger and R. Hennig (2022). Quantum bridge analytics I: a tutorial on formulating and using QUBO models. *Annals of Operations Research*, Vol.314, pp.141-183.
13. L.P. Yu, C.Y. Chen, C.S. Lai, B. Sheu, S.K. Kao and C.R. Chang (2021). Annealing in the noisy intermediate-scale quantum era: key concepts and approaches. *IEEE Nanotechnology Magazine*, Vol.15, No.6, pp.21-27, 2021.
14. G. Kochenberger, J.K. Hao, F. Glover, M. Lewis, Z. Lu, H. Wang and Y. Wang (2014). The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, Vol.28, Issue 1, pp.58-81.
15. C.H. Ou, C.Y. Chen, C.F. Wang and C.R. Chang (2023). Quantum-inspired vehicle routing scheme for rebalancing in bike sharing systems, Vol.13, No.2, pp.2340018-1-8, SPIN.
16. R. Nikouei, N. Rasoulib, S. Tahmasebic, S. Zolfid and H. Faragardie (2019). A quantum-annealing-based approach to optimize the deployment cost of a multi-sink multi-controller WSN. *Procedia Computer Science*, Vol.155, pp.250-257.
17. T. Krauss and J. Mccollum (2020). Solving the network shortest path problem on a quantum annealer. *IEEE Transactions on Quantum Engineering*, Vol.1, pp.1-12.
18. M.R. Garey and D.H. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company.
19. A.H. Karbasi and R.E. Atani (2013). Application of dominating sets in wireless sensor network. *International Journal of Security and Its Applications*, Vol.7, No.4, pp.185-202.
20. J. Yu, N. Wang, G. Wang and D. Yu (2013). Connected dominating sets in wireless ad hoc and sensor networks-A comprehensive survey. *Computer Communications*, Vol.36, Issue 2, pp.121-134.
21. D.Z. Du and P.J. Wan (2013). *Connected Dominating Set: Theory and Applications*, Springer.
22. Z. Zhang, J. Zhou and D. Du (2016). Performance guaranteed approximation algorithm for fault-tolerant connected dominating set in wireless sensor networks*, in Proceeding of the 35th Annual IEEE International Conference on Computer Communications (IEEE INFORCOM'16)*, pp.1-8.
23. L. Song, C. Liu, H. Huang, H. Du and X. Jia (2019). Minimum connected dominating set under routing cost constraint in wireless sensor networs with different transmission ranges. *IEEE/ACM Transactions on Networking*, Vol.27, No.2, pp.546-559.
24. H. Tang, T.C.E. Cheng, and C.T. Ng (2009). Finite dominating sets for the multi-facility ordered median problem in networks and algorithmic applications. *Computers & Industrial Engineering*, Vol.57, Issue 3, pp.707-71.
25. Z. Shao, R. Kosari, R. Anoos, S. Sheikholeslami, J. Dayap and R.L (2020). Gutierrez, Outer-convex dominating set in the corona of graphs as encryption key generator, *Complexity*, 2020.

26. J.R. Jiang and Q.Y. Lin (2023). Utilizing Novel Quantum Counters for Grover's Algorithm to Solve the Dominating Set Problem. arXiv preprint arXiv:2312.09388.
27. A.M. Krol and Z. Al-Ars (2024). Beyond Quantum Shannon: Circuit Construction for General n-Qubit Gates Based on Block ZXZ-Decomposition. arXiv preprint arXiv:2403.13692.
28. M. A. Nielsen and I. L. Chuang (2010). Quantum Computation and Quantum Information, Cambridge University Press.
29. Qiskit (2017). *IBM Quantum*. Version 1.0.0. Available at: https://qiskit.org/.
30. Guifré Vidal (2003). Efficient classical simulation of slightly entangled quantum computations. *Physical review letters*, 91(14):147902.
31. Ulrich Schollwöck (2011). The density-matrix renormalization group in the age of matrix product states. *Annals of physics*, 326(1):96–192.