entropy

MDPI

Article

Solving the Independent Domination Problem by the Quantum Approximate Optimization Algorithm

Haoqian Pan * and Changhong Lu

School of Mathematical Sciences, Key Laboratory of MEA (Ministry of Education) & Shanghai Key Laboratory of PMMP, East China Normal University, Shanghai 200241, China; chlu@math.ecnu.edu.cn
* Correspondence: 52215500040@stu.ecnu.edu.cn

Abstract: In the wake of quantum computing advancements and quantum algorithmic progress, quantum algorithms are increasingly being employed to address a myriad of combinatorial optimization problems. Among these, the Independent Domination Problem (IDP), a derivative of the Domination Problem, has practical implications in various real-world scenarios. Despite this, existing classical algorithms for the IDP are plagued by high computational complexity, and quantum algorithms have yet to tackle this challenge. This paper introduces a Quantum Approximate Optimization Algorithm (QAOA)-based approach to address the IDP. Utilizing IBM's qasm_simulator, we have demonstrated the efficacy of the QAOA in solving the IDP under specific parameter settings, with a computational complexity that surpasses that of classical methods. Our findings offer a novel avenue for the resolution of the IDP.

Keywords: quantum approximate optimization algorithm; independent domination problem; Qiskit

Citation: Pan, H.; Lu, C. Solving the Independent Domination Problem by the Quantum Approximate Optimization Algorithm. *Entropy* **2024**, *26*, 1057. https://doi.org/10.3390/e26121057

Academic Editors: Xiao Yuan, Xiaoming Zhang and Bálint Koczor

Received: 23 October 2024
Revised: 29 November 2024
Accepted: 3 December 2024
Published: 5 December 2024

1. Introduction

The Independent Domination Problem (IDP) seeks to identify the smallest independent dominating set (IDS) within a graph $G(V, E)$. An IDS is defined as a subset $D \subseteq V$ such that every vertex in $V \setminus D$ is adjacent to at least one vertex in D, and no two vertices in D are adjacent. This problem is particularly relevant in practical applications, such as wireless network design. It has been firmly established that the IDP is NP-complete across numerous graph classes, including but not limited to chordal, bipartite, circle, and triangle graphs [1–4]. Due to its inherent computational complexity, research has increasingly focused on approximation and heuristic solutions tailored to specific graph structures, aiming to improve performance and feasibility. Notable advances include a polynomial-time algorithm with a complexity of $O(|V|^2)$ for permutation graphs [5]. Similarly, for cocomparability graphs, a sophisticated algorithm has been proposed with a computational complexity of $O(|V|^{2.376})$, representing a significant optimization for this graph class [6]. Additional efforts have targeted at-most-cubic graphs, showcasing the adaptability of algorithmic approaches to diverse graph types [7]. For general graphs without specific structural constraints, recent work has produced exponential-time algorithms, including those with complexities of $2^{0.529|V|}$, $2^{0.441|V|}$, and $2^{0.424|V|}$, reflecting progress toward more efficient exponential solutions [8–10]. Furthermore, heuristic approaches such as memetic algorithms [11,12] and evolutionary algorithms [13] have gained traction in recent years. Despite these advancements, reducing the algorithmic complexity of the IDP remains a formidable challenge, driving ongoing research and development in this domain.

In recent years, advancements in quantum computing technologies [14–16] have significantly accelerated the application of quantum algorithms to address combinatorial optimization problems. Among these, the Quantum Approximate Optimization Algorithm (QAOA) [17] has emerged as a leading approach, demonstrating versatility across various combinatorial optimization challenges within the quantum computing domain.

A range of problems have been explored using the QAOA on quantum hardware and simulators, including the Max-Cut problem [17], the Traveling Salesman Problem [18], the Domination Problem (DP) [19], the Minimum Vertex Cover Problem [20], the Boolean Satisfiability Problem [21], and the Graph Coloring Problem [22], among others. Beyond these application-specific studies, significant efforts have been devoted to improving the QAOA framework itself. For instance, Chandarana et al. [23] and Wurtz and Love [24] enhanced the convergence rate of the QAOA's approximation ratio by incorporating a problem-dependent counterdiabatic driving term into the QAOA ansatz, effectively reducing the circuit depth. Similarly, Herrman et al. [25] extended the QAOA's rotation angles by introducing vectorized parameters, leading to both increased approximation ratios and reduced quantum circuit depths, particularly for the Max-Cut problem. Chalupnik et al. [26] proposed adding a multi-parameter, problem-independent layer to the QAOA structure, achieving higher approximation ratios when solving Max-Cut on random regular graphs. Furthermore, Wang et al. [27] demonstrated improved performance by selectively optimizing quantum circuits while preserving the integrity of the objective function. Magann et al. [28,29] introduced two algorithms: FALQON and FALQON+. The former leverages qubit measurements in feedback-based quantum optimization to eliminate the need for classical optimizers, while the latter combines FALQON's initialization with the QAOA to enhance parameter initialization. Additionally, Zhou et al. [30] addressed hardware constraints, such as the limited number of qubits on current quantum devices, by proposing the QAOA-in-QAOA method. This approach applies a divide-and-conquer strategy to partition a graph into smaller subgraphs, solving the Max-Cut problem on these subgraphs in parallel using the QAOA and subsequently combining the results with an overarching QAOA layer. This innovation enables the QAOA to tackle larger-scale problems even within the constraints of limited quantum resources. Further recent developments in the QAOA can be found in Blekos et al. [31]. Despite the extensive progress in quantum computing and QAOA research, a notable gap exists in the application of quantum algorithms to the IDP. While the QAOA has proven effective for various combinatorial optimization problems, its potential for addressing the IDP remains unexplored. Thus, the feasibility and effectiveness of the QAOA or other quantum approaches in solving the IDP remain open research questions, warranting further investigation.

The primary contribution of this paper lies in our pioneering application of the QAOA to solve the IDP, marking the first instance of utilizing a quantum algorithm for this specific challenge. We conducted comprehensive evaluations of the effectiveness of the QAOA in addressing the IDP, focusing on two key aspects: fundamental testing and robustness testing. The findings presented herein offer valuable insights and foundational experience that will inform and guide future endeavors aimed at solving the IDP using quantum computing technologies.

The structure of this paper is organized as follows: In Section 2, we systematically introduce the methodology for transforming the IDP into a 0–1 integer programming model, followed by the formulation of a Quadratic Unconstrained Binary Optimization (QUBO) model and its corresponding Hamiltonian. Section 3 presents an overview of the QAOA and delineates the procedural steps required to apply this algorithm for solving the IDP. Subsequently, in Section 4, we conduct both fundamental and robustness testing to assess the effectiveness of the QAOA-based algorithm in addressing the IDP. This section also includes a complexity analysis of the QAOA. Finally, Section 5 provides a comprehensive summary of the paper.

2. Problem Modeling

When using quantum algorithms for combinatorial optimization problems, a common approach is to model or transform the problem into a QUBO format, which is then mapped to the Hamiltonian of the Ising model. In this chapter, we formally define the IDP and derive its corresponding Hamiltonian representation.

2.1. Problem → 0–1 Integer Programming Model

Before exploring the IDP, we first define the dominating set (DS). Given a graph $G = (V, E)$, a DS, D, is a subset of V such that for every vertex $v \in V$, the closed neighborhood $N[v]$ intersects with D (i.e., $N[v] \cap D \neq \emptyset$). Here, $N[v]$ denotes the closed neighborhood of vertex v, comprising v and all vertices adjacent to it. The goal of the DP is to find the smallest such D. The IDP extends the DP by adding a constraint: D must also be independent, meaning no edges exist between vertices in $G[D]$. To enable the application of the QAOA for solving this problem, we first construct a mathematical model of the IDP in the following sections.

$$\min_{\{X_i\}} \quad \sum_{i=1}^{|V|} X_i \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in N[i]} X_j \geq 1 \quad \forall i \in V \tag{2}$$

$$X_i \in \{0, 1\} \quad \forall i \in V \tag{3}$$

$$X_i \cdot X_j = 0 \quad \forall ij \in E \tag{4}$$

In this model, X_i is a binary variable, where $X_i = 1$ indicates that vertex i is included in D, and $X_i = 0$ otherwise. The expression $\sum_{i=1}^{|V|} X_i$ thus represents the total number of vertices in the DS. Our objective is to minimize $|D|$, mathematically expressed as $\sum_{i=1}^{|V|} X_i$. Constraint (2) ensures that for each vertex $v \in V$, either v itself or one of its neighbors is included in D, which is a fundamental requirement of the DP. Constraint (4), specific to the IDP, addresses the edges in G, categorized as follows: (1) edges within $G[D]$, (2) edges within $G[V \setminus D]$, and (3) edges connecting D to $V \setminus D$. For the second and third categories, Constraint (4) is automatically satisfied since at least one vertex in each edge has $X_i = 0$. However, for edges within $G[D]$, Constraint (4) requires that no edges exist, ensuring that the independence condition is met. At this stage, we have established the 0–1 integer programming model for the IDP. The next step is to transform this model into a QUBO formulation and subsequently convert the QUBO model into its Hamiltonian representation.

2.2. 0–1 Integer Programming Model → QUBO Model

The QUBO model is formulated in Equation (5), where x represents the vector of binary variables, and Q is the matrix of constants, commonly known as the QUBO matrix.

$$minimize/maximize \quad y = x^t Q x \tag{5}$$

Since the original objective function already satisfies the requirements of the QUBO model, our primary task at this stage is to convert the constraints into quadratic penalty terms and integrate them into the objective function. For each vertex $i \in V$, Constraint (2) can be expressed in the following general form:

$$X_1 + X_2 + \cdots + X_n \geq 1, \quad n = |N[i]| \tag{6}$$

According to Glover et al. [32], for $n = 1$, the constraint is given by $P \cdot (X_1 - 1)^2$, where P is the penalty coefficient. For $n = 2$, the constraint becomes $P \cdot (1 - X_1 - X_2 - X_1 \cdot X_2)$. For $n \geq 3$, the inequality constraint must be converted into an equality constraint by introducing slack variables, as shown in Equation (7).

$$X_1 + X_2 + \cdots + X_n - S - 1 = 0 \tag{7}$$

It is clear that S lies within the range $[0, n-1]$. We now express S as a combination of binary (0–1) variables. Since S can take any integer value within $[0, n-1]$, it can be represented as follows:

$$S = \sum_{i=1}^{bl_{n-1}-1} X'_i \cdot 2^{i-1} + (n-1-\sum_{i=1}^{bl_{n-1}-1} 2^{i-1}) \cdot X'_{bl_{n-1}} \tag{8}$$

The variable $X'_* \in \{0,1\}$ is an additional binary variable introduced to represent slack variables, and bl_n denotes the binary length of n. By substituting the expression from Equation (8) into Equation (7) and squaring the entire equation, we obtain the quadratic penalty. For Constraint (4), it can be represented as $P \cdot (X_i \cdot X_j)$. The QUBO model for the IDP is presented in Equation (9). In Equation (9), the vertices in V are partitioned based on $|N[i]| = 1, 2$, and $|N[i]| \geq 3$, which allows for the separate treatment of Constraint (2).

$$\begin{aligned}
&\min_{\{X,X'\}} \quad \sum_{i=1}^{|V|} X_i \\
&+ \sum_{i\in V, |N[i]|\geq 3} P \cdot [\sum_{j\in N[i]} X_j - (\sum_{i=1}^{bl_{|N[i]|-1}-1} X'_i \cdot 2^{i-1} + (|N[i]| - 1 - \sum_{i=1}^{bl_{|N[i]|-1}-1} 2^{i-1}) \cdot X'_{bl_{|N[i]|-1}}) - 1]^2 \\
&+ \sum_{i\in V, |N[i]|=2, N[i]=\{j,k\}} P \cdot (1 - X_j - X_k - X_j \cdot X_k) \\
&+ \sum_{i\in V, |N[i]|=1, N[i]=\{j\}} P \cdot (X_j - 1)^2 \\
&+ \sum_{ij\in E} P \cdot X_i \cdot X_j
\end{aligned} \tag{9}$$

2.3. QUBO Model → Hamiltonian

To convert the objective function in Equation (9) into a Hamiltonian, we substitute the binary variables X_i with new variables $s_i \in \{-1, 1\}$. The substitution process is as follows:

$$X_i = \frac{s_i + 1}{2} \tag{10}$$

Then, we have

$$\begin{aligned}
&\min_{\{s,s'\}} \quad \sum_{i=1}^{|V|} \frac{s_i+1}{2} \\
&+ \sum_{i\in V, |N[i]|\geq 3} P \cdot [\sum_{j\in N[i]} \frac{s_j+1}{2} \\
&\quad - (\sum_{i=1}^{bl_{|N[i]|-1}-1} \frac{s'_i+1}{2} \cdot 2^{i-1} + (|N[i]| - 1 - \sum_{i=1}^{bl_{|N[i]|-1}-1} 2^{i-1}) \cdot \frac{s'_{bl_{|N[i]|-1}}+1}{2}) - 1]^2 \\
&+ \sum_{i\in V, |N[i]|=2, N[i]=\{j,k\}} P \cdot (1 - \frac{s_j+1}{2} - \frac{s_k+1}{2} - \frac{s_j+1}{2} \cdot \frac{s_k+1}{2}) \\
&+ \sum_{i\in V, |N[i]|=1, N[i]=\{j\}} P \cdot (\frac{s_j+1}{2} - 1)^2 \\
&+ \sum_{ij\in E} P \cdot \frac{s_i+1}{2} \cdot \frac{s_j+1}{2}
\end{aligned} \tag{11}$$

By replacing s and s' with the Pauli-Z operator σ^z, we ultimately obtain the Hamiltonian.

$$
\begin{aligned}
H_c = & \sum_{i=1}^{|V|} \frac{\sigma_i^z + 1}{2} \\
& + \sum_{i \in V, |N[i]| \geq 3} P \cdot [\sum_{j \in N[i]} \frac{\sigma_j^z + 1}{2} \\
& - (\sum_{i=1}^{bl_{|N[i]|-1}-1} \frac{(\sigma_i^z)' + 1}{2} \cdot 2^{i-1} + (|N[i]| - 1 - \sum_{i=1}^{bl_{|N[i]|-1}-1} 2^{i-1}) \cdot \frac{(\sigma^z_{bl_{|N[i]|-1}})' + 1}{2}) - 1]^2 \\
& + \sum_{i \in V, |N[i]|=2, N[i]=\{j,k\}} P \cdot (1 - \frac{\sigma_j^z + 1}{2} - \frac{\sigma_k^z + 1}{2} - \frac{\sigma_j^z + 1}{2} \cdot \frac{\sigma_k^z + 1}{2}) \\
& + \sum_{i \in V, |N[i]|=1, N[i]=\{j\}} P \cdot (\frac{\sigma_j^z + 1}{2} - 1)^2 \\
& + \sum_{ij \in E} P \cdot \frac{\sigma_i^z + 1}{2} \cdot \frac{\sigma_j^z + 1}{2}
\end{aligned}
\tag{12}
$$

3. QAOA

In this section, we begin by introducing the fundamental concepts of the QAOA. For a quantum system comprising n qubits, $|z\rangle$ denotes a state vector in the 2^n-dimensional Hilbert space, where the bit string z is expressed as $z = z_0 z_1 z_2 \ldots z_{n-1}$. The QAOA starts by applying the Hadamard gate to prepare the initial state $|s\rangle$ from the state $|\underbrace{00\ldots0}_{n}\rangle$.

$$
|s\rangle = \underbrace{\hat{H} \otimes \hat{H} \cdots \otimes \hat{H}}_{n} |\underbrace{00\ldots0}_{n}\rangle = \frac{1}{\sqrt{2^n}} \cdot \sum_z |z\rangle \tag{13}
$$

Then, we use two parametric unitary transformations, $U(C, \gamma)$ and $U(B, \beta)$,

$$
\begin{aligned}
U(C, \gamma) &= e^{-i\gamma H_c} \\
U(B, \beta) &= e^{-i\beta B}
\end{aligned}
$$

where $C = H_c$, $B = \sum_{j=1}^{n} \sigma_j^x$, $\gamma \in [0, 2\pi]$, and $\beta \in [0, \pi]$. By applying these two types of unitary transformations to the initial state $|s\rangle$ and repeating the process q times, we arrive at the final state $|\gamma, \beta\rangle$, as described in Equation (14). In this paper, q is also referred to as the layer number of the QAOA.

$$
|\gamma, \beta\rangle = U(B, \beta_q) U(C, \gamma_q) \cdots U(B, \beta_1) U(C, \gamma_1) |s\rangle \tag{14}
$$

After obtaining $|\gamma, \beta\rangle$, we can measure the expectation value of H_c.

$$
F_q(\gamma, \beta) = \langle \gamma, \beta | H_c | \gamma, \beta \rangle \tag{15}
$$

As a reference, Figure 1 shows the quantum circuit for 10 qubits with two layers.

For the IDP in this paper, our task is to find such γ and β with the minimal expectation value of H_c. The QAOA is a hybrid algorithm that combines a classical optimizer with a quantum computer. We can use a classical optimizer to search for such γ and β. Next, we summarize the steps of using the QAOA to solve the IDP:

Step 1: Convert the IDP into a QUBO model.
Step 2: Transform the QUBO model into a Hamiltonian.
Step 3: Set the layer number q and convert the Hamiltonian into a quantum circuit. IBM's Qiskit integrates many tools, such as QAOAAnsatz. We can generate the quantum circuit by inputting the Hamiltonian and the layer number.
Step 4: Initialize γ and β for each layer.

Step 5: Use a classical optimizer, such as COBYLA, to optimize γ and β. The optimization process terminates when the maximum number of iterations is reached or the predefined function tolerance is satisfied, yielding γ_* and β_*.
Step 6: Update the quantum circuit with γ_* and β_*.
Step 7: Perform multiple samplings on the updated quantum circuit and output the bit string z_* with the highest probability.
Step 8: Derive the IDS of the graph based on z_*.

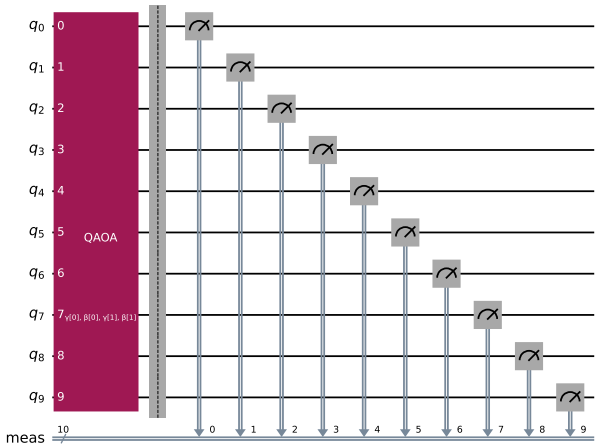

Figure 1. Basic working flow of QAOA with 2 layers and 10 qubits.

4. Experiment

In this section, we evaluate the performance of the QAOA-based algorithm in solving the IDP. The graph used for this analysis is shown in Figure 2. This unweighted graph consists of 6 vertices, with the optimal DS and IDS being distinct. Specifically, the optimal DS is $\{2,3\}$, while the optimal IDS has two possible solutions due to the graph's symmetry: $\{0,4,3\}$ or $\{1,5,2\}$. Selecting a graph where the optimal DS and IDS differ is crucial for assessing the QAOA's ability to solve the IDP rather than the DP. Additionally, the graph's symmetry leads to multiple potential optimal IDSs, which should be reflected in the final probability distribution. Ideally, the two bit strings with the highest probabilities should correspond to the IDSs $\{0,4,3\}$ and $\{1,5,2\}$, with probabilities that are either equal or very close.

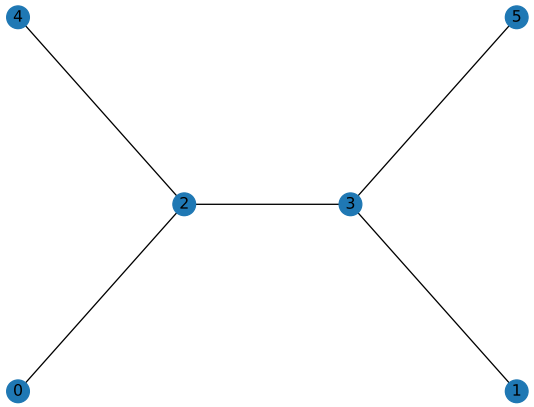

Figure 2. An unweighted graph with 6 nodes and 5 edges.

Based on the modeling method outlined in Section 2, the QUBO formulation of the IDP for this graph is given in Equation (16), which requires 10 qubits. Each term in Equation (16) is explained in Table 1. The corresponding Hamiltonian for this model can be derived by replacing x_* with $\frac{s_*+1}{2}$ and substituting s_* with σ_*^z. For brevity, the

detailed steps of this process are omitted here. Notably, the 10 decision variables in Equation (16) correspond to the 10 qubits in the quantum system. When measuring the energy of this 10-qubit system, the corresponding bit string z will have a length of 10, such as $z = z_0 z_1 \cdots z_5 \cdots z_9$. However, as the optimization model introduces slack variables, we focus primarily on the values of $z_0, z_1, \ldots, z_5$, as these determine which vertices belong to the IDS. For example, when the bit string $z_0 z_1 z_2 z_3 z_4 z_5 = 100110$, it indicates that the set of vertices $\{0, 3, 4\}$ forms an IDS. Thus, when calculating the probability distribution of bit strings, we aggregate the probabilities of all bit strings with identical values for the first 6 qubits. For instance, if the sampling probability is 0.01 for $z = 0000000100$ and 0.02 for $z = 0000001100$, the combined sampling probability for $z = 000000$ would be 0.03. In subsequent sections, when referring to a bit string, we will use the merged 6-bit string representing the first 6 qubits.

$$
\begin{aligned}
\textit{minimize} \quad & x_0 + x_1 + x_2 + x_3 + x_4 + x_5 \\
& + P \cdot (1 - x_0 - x_2 + x_0 x_2) \\
& + P \cdot (1 - x_1 - x_3 + x_1 x_3) \\
& + P \cdot (x_2 + x_0 + x_4 + x_3 - (x_6 + 2x_7) - 1)^2 \\
& + P \cdot (x_3 + x_1 + x_2 + x_5 - (x_8 + 2x_9) - 1)^2 \\
& + P \cdot (1 - x_4 - x_2 + x_4 x_2) \\
& + P \cdot (1 - x_5 - x_3 + x_5 x_3) \\
& + P \cdot x_0 x_2 + P \cdot x_1 x_3 + P \cdot x_2 x_4 + P \cdot x_2 x_3 + P \cdot x_3 x_5
\end{aligned}
\tag{16}
$$

Table 1. Explanation of components of Equation (16).

Component	Description
$x_0 + x_1 + x_2 + x_3 + x_4 + x_5$	Size of IDS.
$P \cdot (1 - x_0 - x_2 + x_0 x_2)$	Penalty for Constraint (2) of node 0 and $N[0] = \{0, 2\}$.
$P \cdot (1 - x_1 - x_3 + x_1 x_3)$	Penalty for Constraint (2) of node 1 and $N[1] = \{1, 3\}$.
$P \cdot (x_2 + x_0 + x_4 + x_3 - (x_6 + 2x_7) - 1)^2$	Penalty for Constraint (2) of node 2 and $N[2] = \{0, 2, 3, 4\}$.
$P \cdot (x_3 + x_1 + x_2 + x_5 - (x_8 + 2x_9) - 1)^2$	Penalty for Constraint (2) of node 3 and $N[3] = \{1, 2, 3, 5\}$.
$P \cdot (1 - x_4 - x_2 + x_4 x_2)$	Penalty for Constraint (2) of node 4 and $N[4] = \{2, 4\}$.
$P \cdot (1 - x_5 - x_3 + x_5 x_3)$	Penalty for Constraint (2) of node 5 and $N[5] = \{3, 5\}$.
$P \cdot x_0 x_2 + P \cdot x_1 x_3 + P \cdot x_2 x_4 + P \cdot x_2 x_3 + P \cdot x_3 x_5$	Penalty for Constraint (4).

In this experiment, we used IBM's qasm_simulator as the backend, with QAOAAnsatz serving as the quantum circuit generator and BaseSamplerV1 as the sampler. The optimization method employed was COBYLA, with a default function tolerance of 10^{-8}. The cost function used for optimization was the Conditional Value-at-Risk (CVaR), which has been widely adopted in various studies to improve both the convergence probability and speed when applied to QAOA [33,34]. As described by Kolotouros and Wallden [33], a variant of CVaR called ascending-CVaR is defined in Equation (17). Here, K represents the total number of samples, and the energy values obtained from the K measurements are sorted in ascending order. $H_{c,i}$ denotes the energy value of the i-th measurement. The cost function is then calculated as the average of the first $\lceil alpha \cdot K \rceil$ energy values. This approach prioritizes lower-energy regions, which are more closely related to the optimal

solution, rather than considering the entire energy distribution, as in Equation (15), thereby accelerating the convergence process.

$$CVaR_{alpha} = \frac{1}{\lceil alpha \cdot K \rceil} \cdot \sum_{i=0}^{\lceil alpha \cdot K \rceil} H_{c,i} \tag{17}$$

COBYLA is a well-known gradient-free optimization algorithm. In each iteration, COBYLA constructs a trust region based on the current CVaR value, using the parameters γ and β. It then selects a set of sample points within this trust region to build a linear approximation model. By solving this linear model, the next iteration point is determined. This process is repeated until the optimal solution is approached. The optimization terminates when the change in the CVaR value is smaller than the function tolerance or when the maximum number of iterations is reached. According to the cost function definition in Equation (17), the convergence of COBYLA is achieved when the change in the average of the first $\lceil alpha \cdot K \rceil$ energy measurements becomes smaller than the function tolerance. For the initial values of γ and β, we adopted the initialization method proposed by Sack and Serbyn [35]. The key parameters considered in the experiment include the number of layers q in the QAOA, the $alpha$ parameter of the CVaR, the penalty coefficient P in the QUBO model, and the maximum number of iterations allowed for COBYLA.

The experiment is divided into two main phases: fundamental testing and robustness testing. In the fundamental testing phase, we assess the performance of the QAOA in solving the IDP using specific parameter settings. The robustness testing phase focuses on comparing the computational results of the QAOA across different parameter configurations. Additionally, this section includes an analysis of the complexity of the QAOA.

4.1. Fundamental Testing

The parameters were set to $q = 15$, $alpha = 0.3$, and $P = 4.5$, where P is 0.75 times the number of nodes in the selected graph, with a maximum of 10,000 iterations. Under these conditions, we employed the QAOA to solve the IDP for the 6-node graph shown in Figure 2. Figure 3 presents the probability distribution of the final sampling results, where the bit strings $z = 011001$ and $z = 100110$ have probabilities of 0.0797 and 0.0793, respectively, significantly higher than those of other configurations. Visualizing these results on the graph, we observe that the outcome is a consequence of the inherent symmetry of the graph (as shown in Figures 4 and 5), demonstrating the effectiveness of the QAOA in solving the IDP. Although the sampling probabilities of the two optimal bit strings are higher than those of other configurations, both remain on the order of 0.01. Given reasonable choices for the penalty coefficients, this may be due to two potential causes. First, a low number of layers might have been used, as suggested by Farhi et al. [17]. Second, the classical optimizer may have become trapped in a local minimum. Since COBYLA is a gradient-free optimization method, it may have disadvantages in escaping local minima compared to gradient-based algorithms like BFGS. Developing a suitable gradient and integrating gradient-based methods, such as BFGS, with the QAOA represents a promising direction for future research.

In Figure 6, we track the cost evolution over the course of the iterations. A sharp decrease in cost is observed within the first $[0, 200]$ iterations, followed by a more gradual decline thereafter. One possible explanation for this behavior is that the penalty term initially outweighs the original objective function. The significant reduction in cost during the $[0, 200]$ interval may be attributed to the optimizer initially focusing on bit strings z that drive the penalty terms in the objective function to zero. For example, in the case of the largest IDS of the graph, $\{0, 1, 4, 5\}$, the corresponding bit string is $z = 110011$. Once the penalty terms are minimized, the optimization shifts toward finding bit strings that minimize the size of the IDS. Although the cost decreases more gradually after 200 iterations, it continues to show a clear downward trend. However, this gradual decline does not alter the fact that $z = 011001$ and $z = 100110$ remain the two bit strings with the highest

probabilities. Further exploration may enhance the probabilities of obtaining correct and optimal results. At the current parameter settings, the probabilities for these bit strings are 0.197 and 0.159, respectively, as shown in Table 2. The calculation of correct and optimal probabilities is as follows: After determining γ_* and β_*, multiple samplings are performed. The correct probability is the sum of the sampling probabilities of the bit strings that satisfy the IDS condition. The optimal probability is the sum of the sampling probabilities of the bit strings whose corresponding vertex sets satisfy the IDS condition and are of optimal size.

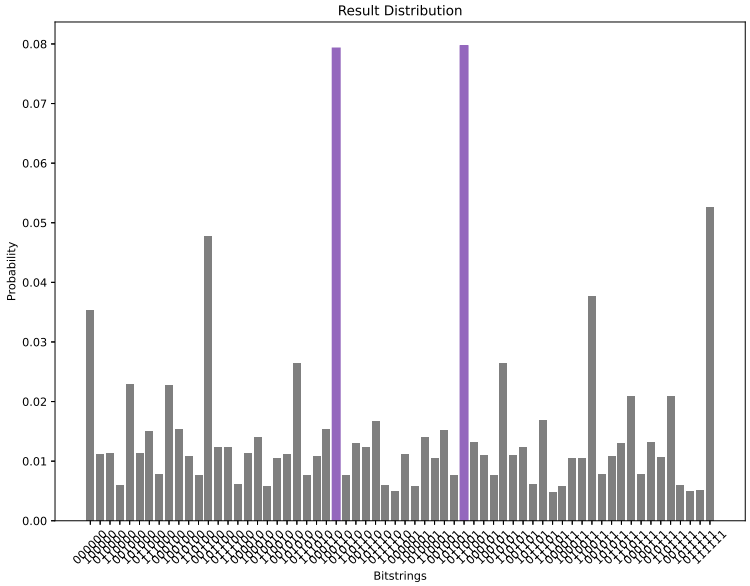

Figure 3. The probability distribution of the bit strings when $q = 15$, $alpha = 0.3$, $P = 4.5$, and maximum iterations = 10,000. The sampling probabilities for the two most probable bit strings are highlighted in purple.

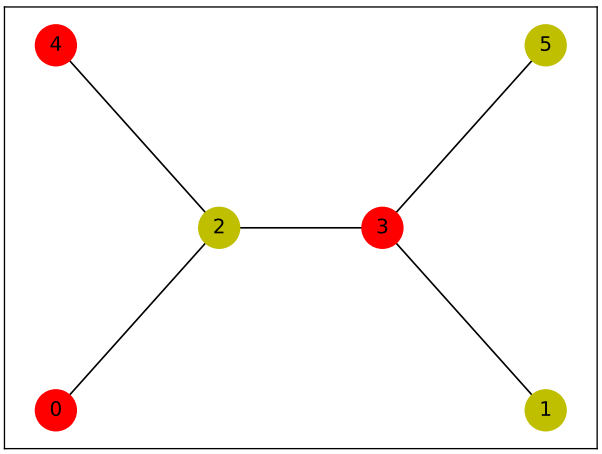

Figure 4. Visualization of the IDS $\{1, 2, 5\}$. The yellow vertices represent the elements of the IDS, while the red vertices indicate the dominated vertices.

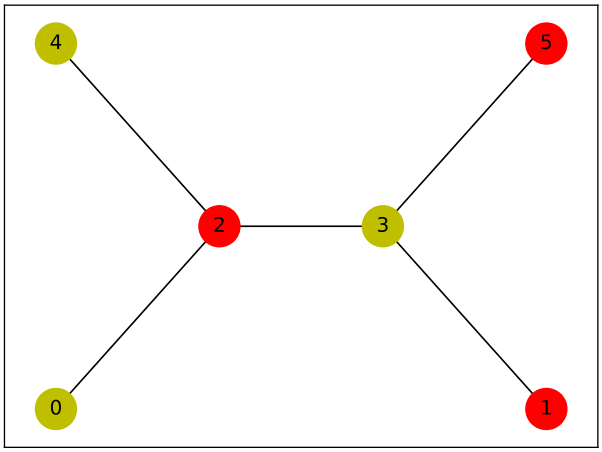

Figure 5. Visualization of the IDS $\{0, 3, 4\}$. The yellow vertices represent the elements of the IDS, while the red vertices indicate the dominated vertices.

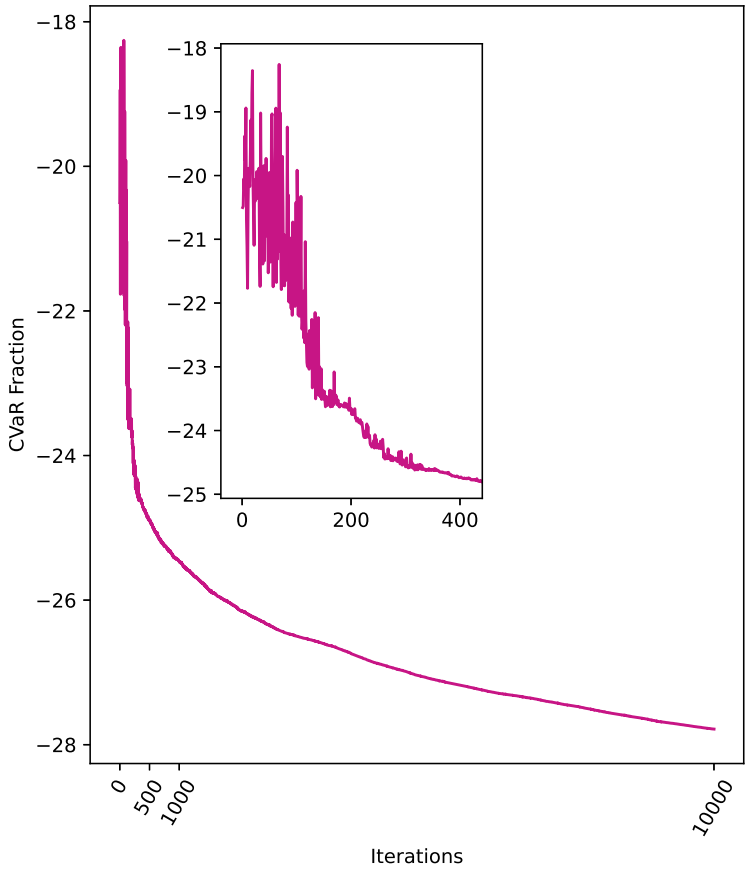

Figure 6. The cost of the QAOA for $q = 15$, $alpha = 0.3$, $P = 4.5$, and a maximum of 10,000 iterations. The zoomed-in region focuses on iterations within the range $[0, 400]$.

Table 2. The values of correct and optimal probabilities when $q = 15$, $alpha = 0.3$, $P = 4.5$, and maximum iterations = 10,000.

Correct	Optimal
0.197	0.159

4.2. Robustness Testing

In this section, we perform a robustness analysis on the four key parameters of the QAOA-based algorithm: $alpha$, q, P, and the maximum number of iterations. A total of 144 parameter combinations were tested, varying $q \in \{10, 15, 20\}$, $alpha \in \{0.3, 0.5, 0.7\}$, $P \in \{3, 4.5, 6, 9\}$, and the maximum number of iterations $\in \{100, 500, 1000, 10{,}000\}$. The possible values of P correspond to 0.5, 0.75, 1, and 1.5 times $|V|$, where $|V|$ represents the upper bound of $|IDS|$. This range is inspired by the work of Glover et al. [32], who recommend setting P between 0.75 and 1.5 times the original objective function. In addition to these values, we have also included $0.5 \cdot |V|$ as a further option. Out of the tested parameter combinations, 97 produced a z_* that satisfied the IDS condition, with 22 of these combinations yielding the optimal IDS. This observation indicates that the effectiveness of the QAOA in solving the IDP is highly dependent on the choice of parameters. By analyzing the performance of the QAOA under different parameter settings, we can better inform the parameter tuning process for future research on using the QAOA to solve the IDP.

In Figure 7, we set $q = 15$, $P = 4.5$, and a maximum of 10,000 iterations and compare the changes in cost for $alpha = 0.3$, 0.5, and 0.7. As observed in Figure 6, the cost decreases sharply within the first $[0, 200]$ iterations for all values of $alpha$. Beyond this range, the reduction in cost becomes more gradual. Overall, the cost reduction trend is consistent across the different values of $alpha$. The data in Figure 7 indicate that the QAOA achieves good convergence when solving the IDP, regardless of the $alpha$ setting. In Figure 8, we compare the cost convergence for $q = 10, 15, 20$, with $P = 4.5$, $alpha = 4.5$, and a maximum of 10,000 iterations. It is evident that when $q = 10$, the cost converges before reaching the maximum number of iterations, meeting the function tolerance. For $q = 15$ and $q = 20$, while the cost quickly levels off for $q = 20$, the cost for $q = 15$ continues to decrease significantly, even at 10,000 iterations. This suggests that, had the maximum number of iterations been further increased, the cost for $q = 15$ would likely continue to decrease. From the perspective of the QAOA, as the number of layers increases, $F_q(\gamma, \beta)$ tends to approach the optimal objective function value [17]. However, for COBYLA, a larger number of layers may negatively affect convergence speed and increase the likelihood of getting trapped in a local minimum. Referring to Figure 8, while the cost converges the fastest when $q = 10$, the CVaR value at convergence is the largest, likely due to the limitation imposed by the smaller number of layers. For $q = 20$, despite having the smallest CVaR value within the set range of maximum iterations, it is possible that the optimization prematurely converges to a local minimum. Therefore, when applying the QAOA to solve the IDP, careful consideration must be given to the choice of the number of layers. A basic recommendation is that when fast convergence is prioritized, starting with a small-layer QAOA structure is advisable. However, when seeking higher-quality solutions, a slightly larger-layer QAOA structure is necessary. In the case of larger-layer QAOA structures, caution should be exercised regarding the potential for getting trapped in a local minimum.

Next, in Figures 9 and 10, we present the optimal and correct probabilities for different combinations of q, P, and maximum iterations, with $alpha = 0.3$. On a macro level, both the optimal and correct probabilities generally increase as the number of maximum iterations increases. When the maximum iteration number is set to 100, many parameter combinations result in low probabilities for both categories, but these probabilities improve significantly as the number of iterations increases. For instance, Table 3 shows how these probabilities change as the number of iterations increases, with $q = 15$, $alpha = 0.3$, and $P = 4.5$. This behavior aligns with our expectations. Additionally, based on observations from Figures 9 and 10, we note that when the maximum iteration number is small, the small-layer QAOA structure outperforms the large-layer QAOA structure in terms of both types of probabilities. This finding contradicts the theoretical prediction that the expectation of H_c should approach the optimal value as the number of layers increases. Upon further analysis, we hypothesize that this behavior arises because the larger-layer QAOA requires optimizing a greater number of angles compared to the smaller-layer

QAOA, as the number of angles is twice the number of layers. For classical optimization algorithms, as the number of angles increases, if the number of iterations is insufficient, the classical optimizer becomes a bottleneck in the process. As the number of iterations increases, the large-layer QAOA structure gradually begins to outperform the small-layer structure in terms of both types of probabilities. Although the large-layer QAOA does not always outperform the small-layer structure for all parameter combinations (for example, in Figure 9, where maximum iterations $= 1000$, $q = 15$, and $P = 3$), the robustness of the tests allows us to observe this trend in most other combinations. From the analysis of Figures 9 and 10, we draw the following insights: when aiming to solve the problem with a smaller maximum number of iterations, a smaller-layer QAOA structure can be used. However, when seeking to maximize the probability of obtaining an IDS or optimal IDS, a larger-layer QAOA structure should be prioritized.

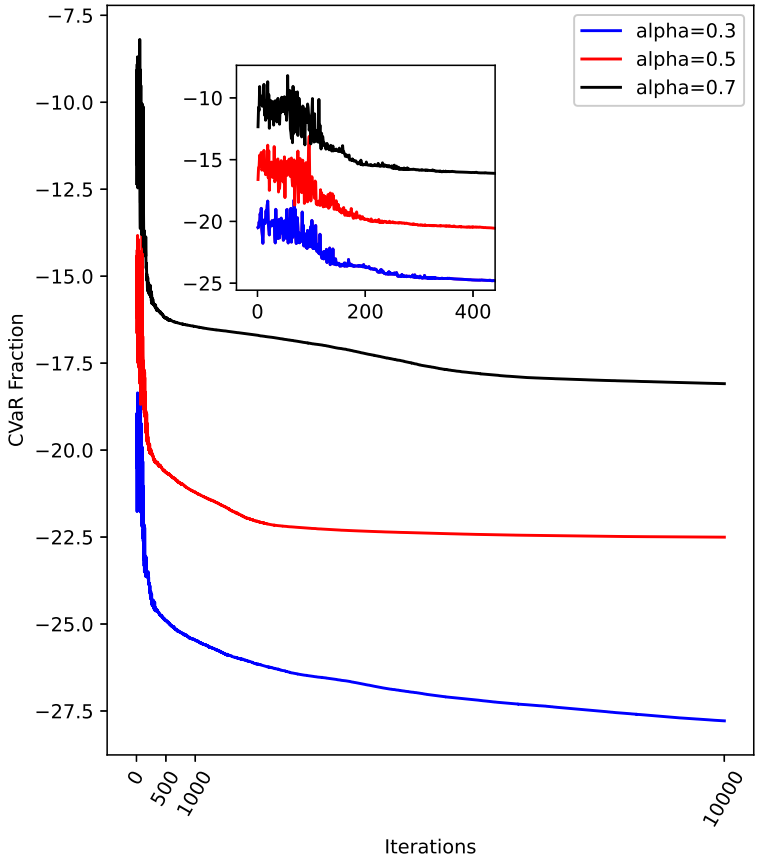

Figure 7. A comparison of the cost for $alpha = 0.3$, 0.5, and 0.7. The values of q, P, and the maximum number of iterations are set to 15, 4.5, and 10,000, respectively. The zoomed-in region focuses on iterations within the range $[0, 400]$.

In Table 4, we compare the impact of different penalty coefficients on the correct and optimal probabilities. The four values of P correspond to 0.5, 0.75, 1, and 1.5 times the number of vertices, which serves as an upper bound for the size of the IDS. These values reflect varying proportions between the penalty term and the original objective function. Our analysis reveals no clear correlation between P and the two probability metrics. Furthermore, even for two closely related penalty coefficients, such as $P = 3$ and $P = 4.5$, we observe significant differences in their corresponding correct and optimal probabilities. This highlights the importance of carefully selecting P, as emphasized by Glover et al. [32]. According to Glover et al. [32], it is recommended to set P to between 0.75 and 1.5 times the original objective function, which aligns with our experimental findings. Additionally, in Figures 9 and 10, when the number of maximum iterations is sufficiently large, both types of probabilities show no significant shortcomings within the

weight range recommended by Glover et al. [32]. This further supports the reasonableness of the chosen range for the penalty coefficient. However, it is important to note that we have not conducted extensive testing outside the range recommended by Glover et al. [32]. Therefore, the results related to the penalty coefficient in this paper serve primarily as a validation of the recommendations made by Glover et al. [32].

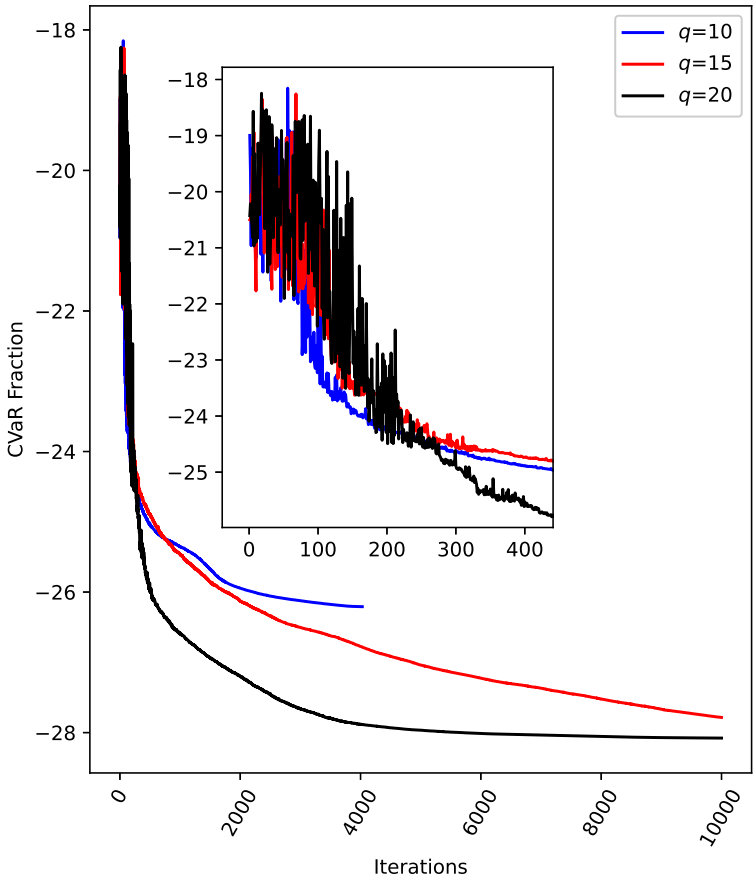

Figure 8. A comparison of the cost for $q = 10$, 15, and 20. The values of p, $alpha$, and the maximum number of iterations are set to 4.5, 0.3, and 10,000, respectively. The zoomed-in region focuses on iterations within the range $[0, 400]$.

Table 3. The comparison of correct and optimal probabilities of different maximum iterations when $q = 15$, $alpha = 0.3$, and $P = 4.5$.

Probability / Maximum Iterations	Correct	Optimal
100	0.100	0.057
500	0.144	0.088
1000	0.149	0.094
10,000	0.197	0.159

As part of the further analysis of the parameter dependency of the QAOA, we recorded the parameter distribution results for all experiments where z_* corresponds to the optimal IDS or an IDS. Taking the number of layers q as an example, in the 144 experiments, z_* was the IDS in 97 of them. Suppose that in these 97 experiments, the number of experiments with $q = 10$, $q = 15$, and $q = 20$ is n_1, n_2, and n_3, respectively. The corresponding ratios are $\frac{n_1}{97}$, $\frac{n_2}{97}$, and $\frac{n_3}{97}$. Similarly, for $alpha$, P, and maximum iterations, we compute the ratios in

the same manner and present the results in pie charts. The statistical results for z_* being the optimal IDS or an IDS are shown in Figures 11 and 12.

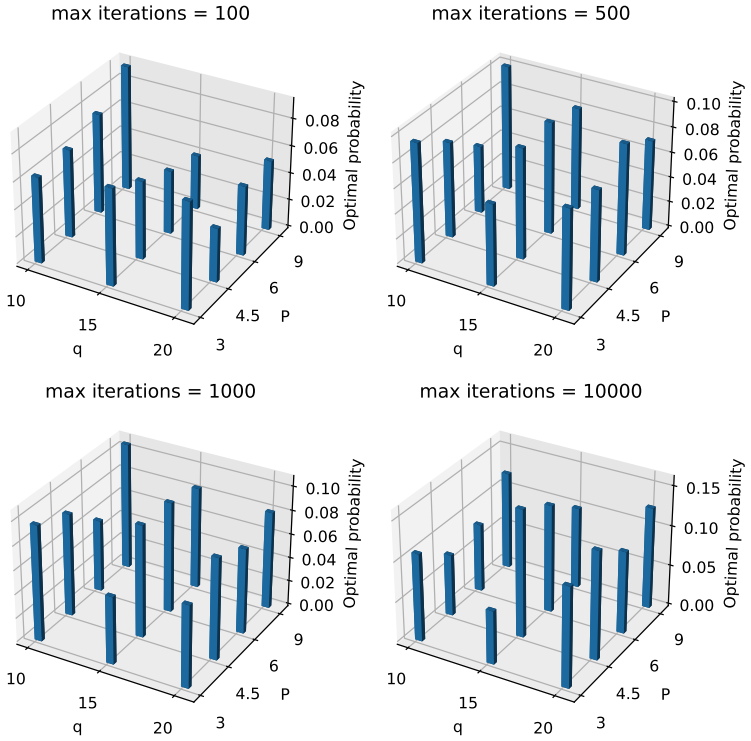

Figure 9. Landscape of optimal probabilities of different q, P, and maximum iterations when $alpha = 0.3$.

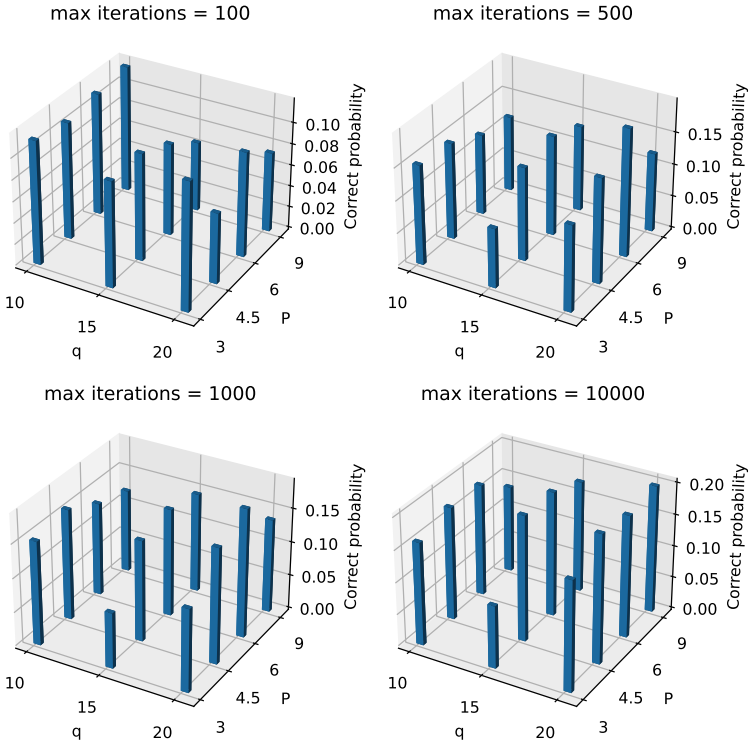

Figure 10. Landscape of correct probabilities of different q, P, and maximum iterations when $alpha = 0.3$.

Table 4. The comparison of correct and optimal probabilities of different punishment coefficients when $q = 15$, $alpha = 0.3$, and maximum iterations = 10,000.

Probability / Punishment Coefficient	Correct	Optimal
$P = 3$	0.098	0.067
$P = 4.5$	0.197	0.159
$P = 6$	0.196	0.135
$P = 9$	0.176	0.102

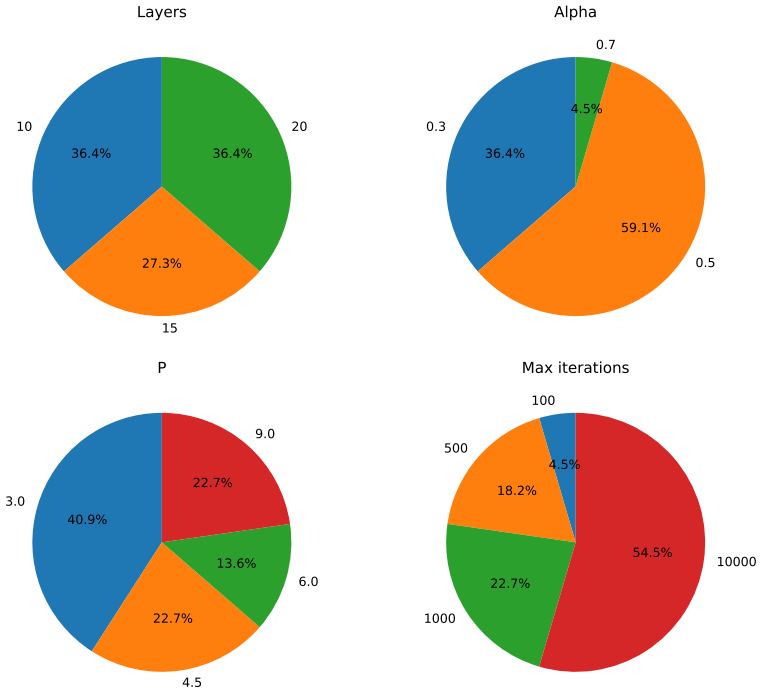

Figure 11. Parameter distribution of experiments in which z_* is an optimal IDS.

Before analyzing the results, it is important to note that the focus of this statistical analysis differs from that of Figures 9 and 10. For example, when z_* is the optimal IDS, the corresponding bit string has the highest sampling probability globally, but this advantage may be small. In other words, the sampling probability of the optimal IDS may not be large, as long as it is slightly higher than that of other bit strings. In contrast, Figure 9 reports the sampling probabilities for the bit string corresponding to the optimal IDS. There may be cases where the optimal probability is large, but z_* is not the optimal IDS. When using QAOA in practice, we can directly consider z_* as the final result, in which case we are primarily concerned with whether z_* is the optimal IDS. Alternatively, we may filter the sampling results and focus more on the size of the optimal probability.

After clarifying the distinction between these two types of metrics, we now turn to the analysis of Figures 11 and 12. We observe that for both z_* being the optimal IDS and an IDS, there is a clear dependency on the maximum number of iterations, while no significant trend is observed for the number of layers. While the small-layer QAOA may be constrained by theoretical performance limitations, it offers the advantage of better convergence. As a result, the small-layer QAOA can still produce an optimal PDS, although the probability of obtaining the optimal PDS may not be very high. This implies that if our goal is to obtain optimal results from the QAOA without focusing on maximizing the optimal probability, we need not excessively increase the number of layers. Instead, we should focus on using a larger maximum number of iterations. Regarding the penalty coefficient P, the results in

Figures 11 and 12 suggest a potential direction for further exploration. Specifically, P values smaller than the range recommended by Glover et al. [32] (e.g., $P = 0.5 \cdot |V|$) might still yield good results. Based on our observations of Equation (16), we find that the penalty term contains 11 terms, and excessively large values of P could cause a large discrepancy between the penalty term and the original objective function, potentially affecting the convergence of the cost. According to the statistical results for P in Figure 11, we hypothesize that slightly reducing P from its current setting might still yield good results. For $alpha$, Figure 11 shows that $alpha = 0.5$ occupies a significantly higher proportion, while no such trend is observed in Figure 12. Given the small number of experiments where z_* is the optimal IDS compared to those where z_* is the IDS, we cannot conclusively determine whether the observed difference is due to statistical fluctuations. Since z_* being the optimal IDS implies that z_* is first an IDS, and the statistical sample for the optimal IDS is relatively small, we are inclined to accept the statistical results for the IDS. Based on these considerations, we believe further testing is needed to better understand the dependency of z_* being the optimal IDS on $alpha$.

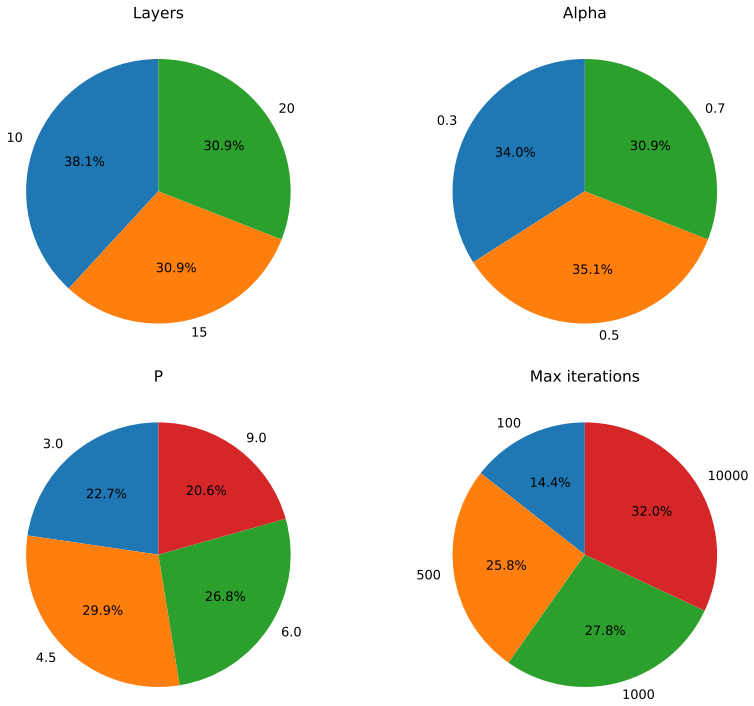

Figure 12. Parameter distribution of experiments in which z_* is an IDS.

4.3. Complexity Analysis

Zhang et al. [20] analyzed the time complexity of the QAOA, which consists of two components. The first component, $O[\text{poly}(q)]$, represents the time complexity associated with the multi-layer structure of the QAOA, which is the most time-consuming aspect. The second component pertains to the time complexity of the optimization algorithm used. In this paper, we employ COBYLA as the optimization algorithm, which has a time complexity of $O[m]$, where m represents the maximum number of iterations. Therefore, the overall time complexity of the QAOA can be expressed as $O[\text{poly}(q) + \text{poly}(m)]$.

In Table 5, we compare the time complexity of the QAOA with other algorithms previously used to solve the IDP. Our analysis shows that the QAOA offers a significant advantage in terms of time complexity when solving the IDP and can be executed on a quantum computer. However, this does not imply that the QAOA is superior in all aspects for solving the IDP. Based on the experimental results and analyses presented in Sections 4.1 and 4.2, we conclude that the QAOA is effective for solving the IDP, provided that the algorithm's parameters are carefully selected. Additionally, there is still room

for improvement in both the accuracy of the results and the probability of obtaining the optimal solution.

Additionally, when executing the QAOA on classical computers, it is important to consider memory constraints. This requirement arises because, during the sampling process for a combinatorial optimization problem involving n qubits, a matrix of size 2^n by the number of samples per bit string must be stored locally. As a result, running the QAOA on a local quantum simulator requires substantial memory overhead.

Table 5. The comparison of the time complexity of algorithms for the IDP.

Algorithm	Time Complexity	Support Quantum Computer?
QAOA	$O[poly(q) + poly(m)]$	Yes
Brandstädt and Kratsch [5]	$O[\|V\|^2]$	No
Breu and Kirkpatrick [6]	$O[\|V\|^{2.376}]$	No
Liu et al. [7] (for (k,l)-graphs)	$O[3.3028^k + \|V\|]$	No
Johnson et al. [8]	$O[2^{0.529\|V\|}]$	No
Gaspers and Liedloff [9]	$O[2^{0.441\|V\|}]$	No
Bourgeois et al. [10]	$O[2^{0.424\|V\|}]$	No

5. Conclusions

In this paper, we explored the use of the quantum algorithm QAOA to solve the IDP. We first modeled the IDP as a 0-1 integer programming problem and converted various constraints into corresponding quadratic penalties, which were added to the original objective function, resulting in the QUBO formulation of the IDP. This QUBO model was then transformed into a Hamiltonian, completing the preparation for inputting the IDP into the quantum algorithm. We applied the QAOA to solve the IDP, a hybrid approach that combines quantum computation with classical optimization. By adjusting the γ and β parameters for each layer of the QAOA through classical optimization, we gradually reduced the expectation value of H_c, which corresponds to the objective function of the IDP.

In the experimental section, we conducted both fundamental and robustness testing. In fundamental testing, we examined the probability distribution and cost variation results of the QAOA in solving the IDP with specific parameters. The results showed that the QAOA-based algorithm is effective for solving the IDP, as the final sampling outcomes reflected multiple IDSs due to the inherent symmetry of the graph. Additionally, the cost gradually flattened after sharp fluctuations within a narrow range, indicating the good convergence performance of the QAOA. In robustness testing, we made the following key observations: (1) By comparing the cost variation curves for different values of *alpha* and q, we found that the CVaR-based cost function showed no significant dependency on *alpha* in terms of convergence speed but did exhibit a tendency with respect to the number of layers. This test also suggested that large-layer QAOA structures may be more prone to getting trapped in local minima. (2) For both optimal and correct probabilities, we compared and analyzed their behavior under different parameters. The results indicated that both types of probabilities showed a slight advantage with larger layer numbers and a higher maximum number of iterations. However, when maximum iterations were set to smaller values, the small-layer QAOA structure performed better. (3) By analyzing the proportions of experimental parameters where z_* was an optimal IDS or an IDS (not necessarily optimal), we found that the QAOA showed no significant preference for the number of layers in these indicators. Additionally, by comparing these results with the results from (2), we highlighted the different scenarios in which these two types of indicators are applicable. In the same section, we also performed a complexity analysis. We confirmed that the QAOA-

based algorithm offers advantages in terms of complexity compared to other algorithms for solving the IDP. Based on the results from fundamental testing, robustness testing, and complexity analysis, we conclude that using the QAOA to solve the IDP is effective and offers advantages in complexity. However, the QAOA-based algorithm cannot guarantee correct or optimal solutions for the problem under arbitrary parameter combinations, which remains a limitation of the algorithm.

As a supplementary note to this work, the experiments presented in this paper used 10 qubits. As the number of vertices in the graph increases, the required number of qubits may also increase. Although we did not test graphs requiring 20, 30, or more qubits due to experimental constraints, we can offer some insights into situations involving larger numbers of qubits. First, considering the analysis in Figures 9 and 10 and acknowledging that the theoretical performance of the QAOA is layer-dependent [17], we believe that for large-scale qubit systems, increasing the number of layers would theoretically enable achieving the probability levels demonstrated in this paper for both optimal and correct probabilities. However, in light of the complexity analysis and the parameter tendency analysis from the experimental section, it is likely that large-scale qubit systems with a large-layer QAOA would require more iterations. Additionally, a larger number of qubits would impose significant memory demands on local quantum simulations. More qubits also present a broader challenge for quantum computing in general. We note that Zhou et al. [30] addressed the Max-Cut problem by partitioning the graph and applying the QAOA to solve the problem on each subgraph. We believe that this approach could inspire solutions for solving the IDP with large-scale qubits. The study of graph partitioning, graph symmetry, and their intersection with the IDP will be highly beneficial for the further development of IDP solving on quantum computing platforms.

The limitations of this paper include the following: (1) We did not conduct tests of the QAOA for solving the IDP on a quantum computer. (2) We did not optimize the quantum circuits used in this study; instead, we utilized circuits generated by IBM's Qiskit package.

In the future, based on the work presented in this paper, our research can be extended in the following directions: (1) testing the performance of the QAOA for solving the IDP on real quantum computers; (2) exploring QAOA parameter ranges that are more suitable for the IDP; (3) exploring the relationship between graph partitioning and the IDP; (4) applying the QAOA to solve other variants of the DP, such as the total DP, the perfect DP, and so forth.

Author Contributions: Conceptualization, H.P.; Methodology, H.P. and C.L.; Software, H.P.; Validation, H.P. and C.L.; Formal Analysis, H.P. and C.L.; Investigation, H.P. and C.L.; Resources, C.L.; Data Curation, H.P.; Writing—Original Draft Preparation, H.P.; Writing—Review and Editing, C.L.; Visualization, H.P.; Supervision, C.L.; Project Administration, C.L.; Funding Acquisition, C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (No. 12331014).

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are included within the article.

Acknowledgments: This work was inspired by the GitHub project named qopt-best-practices, which can be accessed at [https://github.com/qiskit-community/qopt-best-practices.git] (accessed on 1 June 2024). Additionally, we express our gratitude to IBM for providing the Qiskit package, which offers essential tools for quantum programming.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DP	Domination Problem
DS	Dominating set
IDP	Independent Domination Problem
IDS	Independent dominating set
QUBO	Quadratic Unconstrained Binary Optimization
QAOA	Quantum Approximate Optimization Algorithm
CVaR	Conditional Value-at-Risk

References

1. Farber, M. Independent domination in chordal graphs. *Oper. Res. Lett.* **1982**, *1*, 134–138. [CrossRef]
2. Irving, R.W. On approximating the minimum independent dominating set. *Inf. Process. Lett.* **1991**, *37*, 197–200. [CrossRef]
3. Aggarwal, A.; Rangan, C.P.; Damian-Iordache, M.; Pemmaraju, S.V. Hardness of approximating independent domination in circle graphs. In Proceedings of the Algorithms and Computation: 10th International Symposium, ISAAC'99, Chennai, India, 16–18 December 1999; Proceedings 10; Springer: Berlin/Heidelberg, Germany, 1999; pp. 56–69.
4. Orlovich, Y.L.; Zverovich, I.E. Independent domination in triangle graphs. *Electron. Notes Discret. Math.* **2007**, *28*, 341–348. [CrossRef]
5. Brandstädt, A.; Kratsch, D. On domination problems for permutation and other graphs. *Theor. Comput. Sci.* **1987**, *54*, 181–198. [CrossRef]
6. Breu, H.; Kirkpatrick, D. Algorithms for dominating and Steiner set problems in cocomparability. Manuscript, 1996. Available online: https://link.springer.com/chapter/10.1007/978-1-4613-0303-9_28 (accessed on 13 October 2024).
7. Liu, C.H.; Poon, S.H.; Lin, J.Y. Independent dominating set problem revisited. *Theor. Comput. Sci.* **2015**, *562*, 1–22. [CrossRef]
8. Johnson, D.S.; Yannakakis, M.; Papadimitriou, C.H. On generating all maximal independent sets. *Inf. Process. Lett.* **1988**, *27*, 119–123. [CrossRef]
9. Gaspers, S.; Liedloff, M. A branch-and-reduce algorithm for finding a minimum independent dominating set in graphs. In Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science, Bergen, Norway, 22–24 June 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 78–89.
10. Bourgeois, N.; Della Croce, F.; Escoffier, B.; Paschos, V.T. Fast algorithms for min independent dominating set. *Discret. Appl. Math.* **2013**, *161*, 558–572. [CrossRef]
11. Wang, Y.; Chen, J.; Sun, H.; Yin, M. A memetic algorithm for minimum independent dominating set problem. *Neural Comput. Appl.* **2018**, *30*, 2519–2529. [CrossRef]
12. Zhou, Y.; Li, J.; Liu, Y.; Lv, S.; Lai, Y.; Wang, J. Improved Memetic Algorithm for Solving the Minimum Weight Vertex Independent Dominating Set. *Mathematics* **2020**, *8*, 1155. [CrossRef]
13. Pan, S.; Ma, Y.; Wang, Y.; Zhou, Z.; Ji, J.; Yin, M.; Hu, S. An improved master-apprentice evolutionary algorithm for minimum independent dominating set problem. *Front. Comput. Sci.* **2023**, *17*, 174326. [CrossRef]
14. Raussendorf, R.; Briegel, H.J. A one-way quantum computer. *Phys. Rev. Lett.* **2001**, *86*, 5188. [CrossRef] [PubMed]
15. Ladd, T.D.; Jelezko, F.; Laflamme, R.; Nakamura, Y.; Monroe, C.; O'Brien, J.L. Quantum computers. *Nature* **2010**, *464*, 45–53. [CrossRef] [PubMed]
16. Sood, S.K. Quantum computing review: A decade of research. *IEEE Trans. Eng. Manag.* **2023**, *71*, 6662–6676. [CrossRef]
17. Farhi, E.; Goldstone, J.; Gutmann, S. A Quantum Approximate Optimization Algorithm. *arXiv* **2014**, arXiv:1411.4028.
18. Radzihovsky, M.; Murphy, J.; Mason, S. A QAOA Solution to the Traveling Salesman Problem Using pyQuil. 2019. Available online: https://cs269q.stanford.edu/projects2019/radzihovsky_murphy_swofford_Y.pdf (accessed on 13 October 2024).
19. Guerrero, N.J. Solving Combinatorial Optimization Problems Using the Quantum Approximation Optimization Algorithm. Ph.D. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, USA, 2020. Available online: https://core.ac.uk/reader/328162090 (accessed on 1 July 2024).
20. Zhang, Y.J.; Mu, X.D.; Liu, X.W.; Wang, X.Y.; Zhang, X.; Li, K.; Wu, T.Y.; Zhao, D.; Dong, C. Applying the quantum approximate optimization algorithm to the minimum vertex cover problem. *Appl. Soft Comput.* **2022**, *118*, 108554. [CrossRef]
21. Yu, Y.; Cao, C.; Wang, X.B.; Shannon, N.; Joynt, R. Solution of SAT problems with the adaptive-bias quantum approximate optimization algorithm. *Phys. Rev. Res.* **2023**, *5*, 023147. [CrossRef]
22. Jansen, D.; Heightman, T.; Mortimer, L.; Perito, I.; Acín, A. Qudit inspired optimization for graph coloring. *arXiv* **2024**, arXiv:2406.00792. [CrossRef]
23. Chandarana, P.; Hegade, N.N.; Paul, K.; Albarrán-Arriagada, F.; Solano, E.; Del Campo, A.; Chen, X. Digitized-counterdiabatic quantum approximate optimization algorithm. *Phys. Rev. Res.* **2022**, *4*, 013141. [CrossRef]
24. Wurtz, J.; Love, P.J. Counterdiabaticity and the quantum approximate optimization algorithm. *Quantum* **2022**, *6*, 635. [CrossRef]
25. Herrman, R.; Lotshaw, P.C.; Ostrowski, J.; Humble, T.S.; Siopsis, G. Multi-angle quantum approximate optimization algorithm. *Sci. Rep.* **2022**, *12*, 6781. [CrossRef]

26. Chalupnik, M.; Melo, H.; Alexeev, Y.; Galda, A. Augmenting QAOA ansatz with multiparameter problem-independent layer. In Proceedings of the 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), Broomfield, CO, USA, 18–23 September 2022; IEEE: New York, NY, USA, 2022; pp. 97–103.
27. Wang, Z.D.; Zheng, P.L.; Wu, B.; Zhang, Y. Quantum dropout for efficient quantum approximate optimization algorithm on combinatorial optimization problems. *arXiv* **2022**, arXiv:2203.10101.
28. Magann, A.B.; Rudinger, K.M.; Grace, M.D.; Sarovar, M. Feedback-based quantum optimization. *Phys. Rev. Lett.* **2022**, *129*, 250502. [CrossRef] [PubMed]
29. Magann, A.B.; Rudinger, K.M.; Grace, M.D.; Sarovar, M. Lyapunov-control-inspired strategies for quantum combinatorial optimization. *Phys. Rev. A* **2022**, *106*, 062414. [CrossRef]
30. Zhou, Z.; Du, Y.; Tian, X.; Tao, D. QAOA-in-QAOA: Solving large-scale MaxCut problems on small quantum machines. *Phys. Rev. Appl.* **2023**, *19*, 024027. [CrossRef]
31. Blekos, K.; Brand, D.; Ceschini, A.; Chou, C.H.; Li, R.H.; Pandya, K.; Summer, A. A review on quantum approximate optimization algorithm and its variants. *Phys. Rep.* **2024**, *1068*, 1–66. [CrossRef]
32. Glover, F.; Kochenberger, G.; Hennig, R.; Du, Y. Quantum bridge analytics I: A tutorial on formulating and using QUBO models. *Ann. Oper. Res.* **2022**, *314*, 141–183. [CrossRef]
33. Kolotouros, I.; Wallden, P. Evolving objective function for improved variational quantum optimization. *Phys. Rev. Res.* **2022**, *4*, 023225. [CrossRef]
34. Yu, Q.; Jin, R. Improved Quantum Approximate Optimization Algorithm Based on Conditional Value-at-Risk for Portfolio Optimization. 2024. Available online: https://assets-eu.researchsquare.com/files/rs-4582391/v1_covered_89dfd8ef-8ea6-40ef-905b-0c7a9f2ba4f7.pdf (accessed on 13 October 2024).
35. Sack, S.H.; Serbyn, M. Quantum annealing initialization of the quantum approximate optimization algorithm. *Quantum* **2021**, *5*, 491. [CrossRef]