Study and Analysis of QAOA

June Vanlerberghe

December 13, 2024

Abstract

This paper introduces the Quantum Approximate Optimization Algorithm, a promising algorithm for solving combinatorial optimization problems using hybrid quantum-classical techniques. QAOA uses the technique of variational quantum algorithms, where the problem is encoded into a Hamiltonian used in the quantum circuit, and the optimization is performed by adjusting the parameters classically. This algorithm has shown potential for finding approximate solutions to QUBO problems, including MaxCut. This paper explores the theoretical, simulated, and experimental performance of QAOA and compares it with existing classical approaches.

1 Introduction

Optimization problems have many important applications in logistics, finance, and machine learning. Classical approaches to combinatorial optimization problems, such as Quadratic Unconstrained Binary Optimization (QUBO) and MaxCut, face significant challenges due to their NP-hard nature. Approximation algorithms have become widely researched, and there is growing interest in how quantum computers can help solve these problems.

Today's quantum hardware is considered Noisy Intermediate-Scale Quantum (NISQ), meaning they have a limited number of qubits and are noisy [22]. Due to this, there is wide interest in developing algorithms that can run on these noisy computers. Variational Quantum Algorithms (VQAs) have emerged as a promising approach for using near-term devices due to their hybrid quantum-classical nature and shallow circuit depth [6].

An example of a VQA is the Quantum Approximate Optimization Algorithm (QAOA), first proposed by Fahri, Goldstone, and Gutmann in 2014 [11]. QAOA finds approximate solutions to combinatorial optimization problems by encoding the problem into a Hamiltonian, which is part of the quantum circuit. The other part of this circuit is a simpler "mixer" Hamiltonian. These Hamiltonians have parameters that are optimized classically. The performance of combinatorial optimization algorithms such as QAOA is measured by the approximation ratio: $\alpha = C_{QAOA}/C_{max}$ where C_{QAOA} is the cost of the solution found by QAOA, and C_{max} is the cost of the optimal solution [3]. Theoretically, this approximation ratio increases with the number of layers in the QAOA circuit, p, making QAOA a promising candidate for optimization problems.

The paper is organized as follows. Section 2 is the Background section, which includes all the necessary information to understand how QAOA works. It will explain the MaxCut problem, QUBO problems, variational quantum algorithms, and the quantum adiabatic algorithm (QAA). Section 3 explains QAOA, with a high-level overview and step-by-step explanation of how the algorithm works. Section 4 is an analysis covering the performance of QAOA (theoretically, simulated, and experimentally) and how it compares to the performance of classical algorithms.

2 Background

Optimization is the act of finding the minimum or maximum of a function, called the objective function [15]. In combinatorial optimization, the domain of this function is typically finite[15]. Although selecting the best solution from finite possibilities may seem trivial, these functions can become extremely complex at even moderate size. When problems become too difficult to find the optimal solution, algorithms can be designed to find approximate solutions [15].

A special case is Quadratic Unconstrained Binary Optimization (QUBO). The goal is to maximize/minimize a quadratic function that takes either 0 or 1 as a variable without other constraints on the variable [23]. The difference between maximization and minimization is simply a flip of the sign of the cost function. This problem can be written as follows:

$$\text{min/maximize: } C(x) = \quad x^T Q x + c^T x \tag{1}$$

$$x \in \{0,1\}^n, Q \in \mathbb{R}^{n \times n}, c \in \mathbb{R}^n$$

QUBO problems are NP-complete, meaning they are an NP problem (nondeterministic polynomial time) and

NP-hard. A problem is said to be NP-hard if an algorithm to solve it can be translated into solving any other NP problem [27]. QUBO can be applied to problems such as MaxCut, traveling salesman, and scheduling [23]. Since they are NP-hard, it is very difficult for a classical algorithm to find the exact or approximate solution. People are looking to leverage quantum techniques such as QAOA to help solve these types of problems.

2.1 MaxCut

MaxCut is a type of problem that can always be formulated as a QUBO and is one of the most difficult combinatorial optimization problems to solve [7]. MaxCut involves a graph with vertices (or nodes) V connected by edges E (that may be weighted). The goal is to find the partition of vertices with the highest weight. The weight of a "cut" is the sum of the weights of the edges that connect the vertices from different sets.

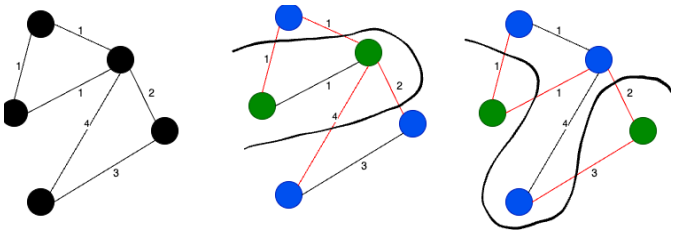

Figure 1: a) Example of a graph for MaxCut. b) Example cut with weight $= 1 + 1 + 2 + 4 = 8$. c) A different cut with weight $= 1 + 1 + 2 + 3 = 7$.

2.1.1 Formulation

Consider an undirected graph G = (V, E), where V is the set of vertices, E is the set of edges, and w_{ij} is the weight of the edge that connects vertex i and j. A cut is the division of vertices into two subsets (0 and 1). The weight of the cut is the sum of the edges that connect vertices of different sets. This weight is represented by the cost function:

$$C(x) = \sum_{i,j=1}^{n} W_{ij} x_i (1 - x_j) \tag{2}$$

2.1.2 MaxCut to QUBO

As explained before, MaxCut can be formulated as a QUBO. Here we reformulate the MaxCut cost function in accordance with QUBO:

$$Q_{ij} = -W_{ij} \qquad c_i = \sum_{j=1}^{n} W_{ij} \tag{3}$$

$$C(x) = \sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} c_i x_i = x^T Q x + c^T x \tag{4}$$

2.1.3 Classical Solutions

Since this is an NP-hard problem, an approximation algorithm is the best solution. First, we should define the approximation ratio:

$$\alpha = \frac{C}{C_{max}} \tag{5}$$

where C is the cost associated with the solution output and C_{max} is the cost associated with the optimal solution. It is known that achieving an approximation ratio higher than $16/17 \approx 0.941$ is an NP-hard problem [17]. The best classical approximation algorithm is the Goemans-Williamson algorithm, which guarantees an approximation ratio of $\alpha \geq 0.878$ using semidefinite programming techniques [14]. Other solutions for more specific graphs use techniques such as greedy randomized search, variable neighborhood search, and path-relinking algorithms [14]. Therefore, leveraging quantum techniques to solve optimization problems such as MaxCut is a highly researched topic.

2.2 Variational Quantum Algorithms

Variational Quantum Algorithms are a broader category of quantum algorithms that include QAOA. These algorithms aim to find the ground state of a Hamiltonian (H) using the variational principle [6]. The variational principle states that the ground state energy is always less than or equal to the expectation value of H calculated with the trial wave function [6].

$$E_0 \leq \langle \psi | H | \psi \rangle \tag{6}$$

So, by varying $|\psi\rangle$ until the expectation value is minimized, we can obtain an approximation for the ground state and its energy. A diagram of how these algorithms work is shown in Fig. 2.

The first step is to encode the problem into a cost function that we can optimize [6]. We have already defined our cost functions for MaxCut, in equations (2) and (4). The minimum of the cost function should correspond to the solution of the problem [6]. The next step is to develop an ansatz, a quantum circuit that will map the input state to the desired ground state. There are many different possible ansatzes depending on the problem. For QAOA, we have a "quantum alternating operator ansatz", which sequentially applies alternating unitaries, U_M and U_C, to get to the ground state [6]. $U_M = e^{-i\beta H_M}$ and $U_C = e^{-i\gamma H_C}$ where the parameterized values are $\theta = (\gamma, \beta)$ [6].

This information is then given to a quantum computer to efficiently find the output of the cost function. This result is given to a classical computer to optimize the parameters. Many methods exist for this optimization, including stochastic gradient descent [6]. Once the parameters are optimized, they are given back to the variational

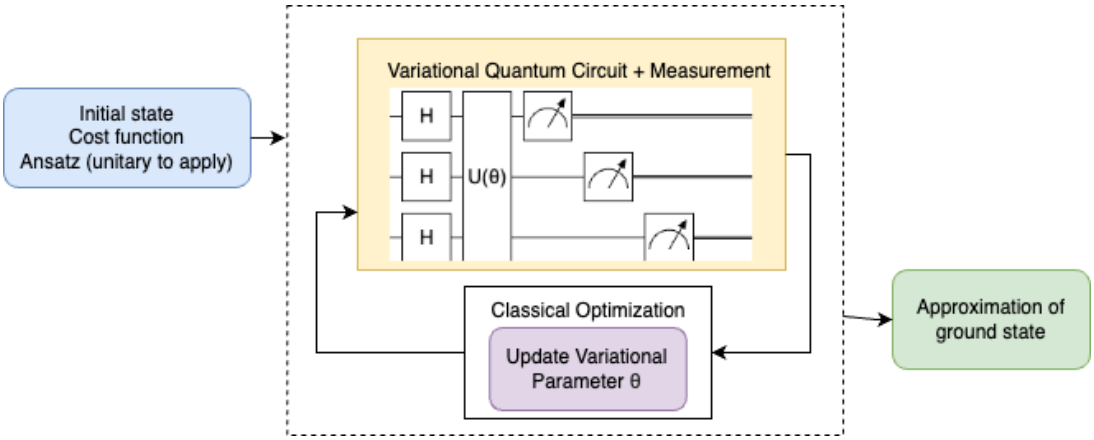

Figure 2: Diagram of variational quantum algorithms

quantum circuit to perform another measurement. This hybrid quantum-classical loop can be repeated many times to reach the desired result.

2.2.1 Variational Quantum Eigensolver

The variational quantum eigensolver was first introduced by Peruzzo et al. [21] and later improved upon by McClean et al. [20]. The objective is to solve for the lowest energy state of a given Hamiltonian by using the variational quantum algorithm techniques described in the previous section. The cost function is defined as:

$$E_{VQE} = min_{\theta} \langle 0|U^{\dagger}(\theta)\hat{H}U(\theta)|0\rangle \tag{7}$$

where the initial state is typically $|0\rangle$ and the ansatz is the parameterized unitary U(θ). The state is then iteratively optimized by varying the parameters θ towards the optimal parameters [26].

This algorithm was proposed as an alternative to phase estimation that reduces the requirements for coherent evolution [21]. Quantum phase estimation does offer an exponential speedup over classical methods, but it requires many quantum operations and thus needs long coherence times and minimal noise [21]. VQEs take advantage of the combination of classical and quantum resources while still providing an exponential speedup over classical algorithms, making it a more useful near-term algorithm than phase estimation.

2.2.2 Quantum Adiabatic Algorithm

The Quantum Adiabatic Algorithm was proposed in 2001 by Fahri et al. [12], which takes advantage of the adiabatic theorem. We know that quantum systems evolve in time according to the Schrodinger equation:

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle \tag{8}$$

where $|\psi(t)\rangle$ is the wave function and H(t) is the time-dependent Hamiltonian. The adiabatic theorem states that if we start in the ground state of a time-dependent Hamiltonian and slowly evolve H, the resulting state $|\psi(t)\rangle$ will be the ground state of the final Hamiltonian [12].

We can start by choosing a Hamiltonian H_S with a simple and known ground state. For any problem H_P with a complex ground state, we can find this state by slowly evolving H_S to H_P [12]. We define H(t) over a time period $0 \leq t \leq T$, where H(0) = H_S and H(T) = H_P [12]. This relates to QAOA because as the number of layers in QAOA increases towards ∞, the algorithm will find the optimal solution according to the adiabatic theorem [11].

2.2.3 Trotterization

If we have a Hamiltonian $H_C = H_1 + H_2$, then we can write the time evolution as follows:

$$U(t) = e^{-iHt/\hbar} = e^{-i(H_1+H_2)t/\hbar} \tag{9}$$

For matrices that commute, we know that $e^{A+B} = e^A e^B$. However, this does not apply to matrices that do not commute. Instead, there exists an approximation called the Trotter Product Formula, which finds that:

$$e^{-i(A+B)t} \approx (e^{-iAt/r}e^{-iBt/r})^r \tag{10}$$

where r is known as the Trotterization depth [18]. The higher r is, the more accurate the approximation will be. This is useful to QAOA as each layer of the algorithm corresponds to trotterized segments of the time evolution operator [11].

3 The Quantum Approximate Optimization Algorithm (QAOA)

The Quantum Approximate Optimization Algorithm (QAOA) was first introduced in 2014 by Fahri, Goldstone, and Gutmann [11]. QAOA is a VQA that generates approximate solutions for combinatorial optimization problems and is said to be suitable for running on near-term devices, due to its shallow circuit depth. QAOA is a trotterized version of QAA, designed to iteratively apply alternating unitaries until the best approximation is found [11]. The unitaries consist of a cost and a mixer where the cost is what the problem is encoded into, and the mixer is a simpler Hamiltonian for which the ground state is known [3]. In contrast to QAA, QAOA has a feature in which as the parameter p increases, the approximation improves [11].

MaxCut is a well-known combinatorial optimization problem that is often used to illustrate QAOA. This is due to its complexity (NP-hard) and its extensive research in classical algorithms. Its cost function is also easily translatable to Hamiltonians, making it a great candidate to demonstrate QAOA's capabilities. This is why the following explanation of QAOA will be to solve the MaxCut problem. The algorithm has the following steps:

1. Define a cost Hamiltonian H_C that encodes the problem and construct the operator U_C. The cost function can be transformed to the cost Hamiltonian as such [3]:

$$\hat{H}_C |x\rangle = C(x) |x\rangle \tag{11}$$

For MaxCut, we have already defined the cost function in (2). The Hamiltonian H_C can be written as:

$$H_C = \sum_{i,j=0}^{n} \frac{w_{ij}}{2}(\mathbb{I} - \sigma_i^Z \sigma_j^Z) \tag{12}$$

Construct the operator U_C [11]:

$$U_C = e^{-i\gamma H_C} \tag{13}$$

2. Construct a second Hamiltonian and operator, H_M, which we call the mixer. This is typically the mixer Hamiltonian used for MaxCut [11]:

$$H_M = \sum_{i=0}^{n} \sigma_i^X \qquad U_M = e^{-i\beta H_B} \tag{14}$$

where σ^X is the Pauli-X operator [11]. Recall that the exponential of a single-qubit unitary can be expressed as a rotation, so we can write U_M as [5]:

$$U_M(\beta) = \prod_{i=1}^{n} R_{X_i}(2\beta) \tag{15}$$

3. Initialize the state in the highest energy state of the mixer, H_M [3]:

$$|\psi_0\rangle = |+\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{16}$$

4. Construct the QAOA circuit. Once the state is initialized, the circuit is created by repeating the QAOA layer p times [11] (see Fig. 3). The QAOA layer is defined as the U_C operator followed by the U_M operator [11]. There are also 2p parameters that need to be defined: γ_{1-p} and β_{1-p} such that $\gamma \in [0, 2*\pi)$ and $\beta \in [0, \pi)$ [3]. The circuit is defined as [11]:

$$|\psi\rangle = \hat{U}_C(\gamma_p)\hat{U}_M(\beta_p)...\hat{U}_C(\gamma_1)\hat{U}_M(\beta_1) |\psi_0\rangle \tag{17}$$

5. Next, find the expectation value of H_C in this state $|\psi\rangle$, which represents the cost obtained for the problem [30]:

$$F_p(\gamma, \beta) = \langle \psi_p(\gamma, \beta) | H_C | \psi_p(\gamma, \beta) \rangle \tag{18}$$

6. A classical computer is then used to optimize the parameters γ and β to maximize the expectation value (cost) [30]. This optimization can be performed in a variety of ways, such as gradient descent optimization [30]. Once the expectation value is maximized, the quality of the approximation is calculated using the approximation ratio (equation 5).

3.1 Application to QAA

Fahri et al. also discuss the connection of QAOA to QAA (see section 2.2.2) [11]. QAOA is focused on finding an approximate solution to the problem, while the Quantum Adiabatic Algorithm (QAA) is designed to find the optimal solution if the Hamiltonian is evolved slowly enough (if the runtime is long enough). Consider the adiabatic evolution with runtime T:

$$H(t) = \frac{t}{T} H_C + (1 - \frac{t}{T}) H_M \tag{19}$$

Where H_M is our mixer Hamiltonian, which defines our initial state as its highest energy state, $|+\rangle$ and H_C is our cost Hamiltonian. A Trotterized approximation to this time evolution corresponds to a layer of QAOA. So, according to the adiabatic theorem, if p is large enough ($p \to \infty$), QAOA would produce the optimal solution.

4 Analysis

4.1 Performance

Fahri et al. demonstrated the performance of QAOA on several types of graphs [11]. A 2-regular graph is a

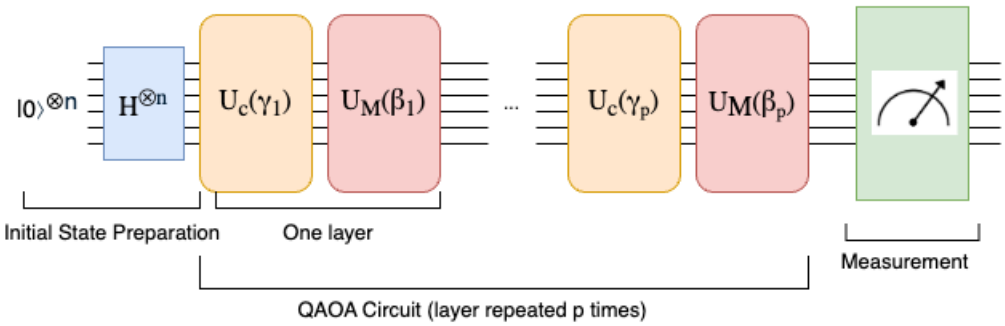

Figure 3: Figure adapted from [3]. Shows an example QAOA circuit with the initial state being prepared as $|++..+\rangle$

graph in which each vertex has two edges and contains one or more (disconnected) cycles [11]. For a 2-regular graph, QAOA is able to provide an approximation ratio: $\alpha \geq (2p+1)/(2p+2)$ [11]. This is significant because this approximation ratio does not depend on the problem size, only on p (number of layers). Moreover, the approximation ratio increases with increasing p, which means that with high enough p, QAOA can find a solution close to the optimal. For a 3-regular graph, Fahri states that for p = 1, the worst-case approximation ratio is: $\alpha = 0.6924$ [11]. More approximation ratios for QAOA are shown in the table (see Table 1) [11, 29, 16]. It becomes more complex to solve for performance guarantees for higher p, which is why Fahri et al. only found the approximation ratio for p = 2.

Graph Type	Algorithm	Parameters	Approx. Ratio
2-regular	QAOA	for any p	$\alpha \geq \frac{(2p+1)}{(2p+2)}$
3-regular	QAOA	p = 1	$\alpha \geq 0.6924$
3-regular	QAOA	p = 2	$\alpha \geq 0.7559$
3-regular	QAOA	p = 3	$\alpha \geq 0.7924$
Any graph	GW	N/A	$\alpha \geq 0.8786$
$\leq$ 3-regular	SDP	N/A	$\alpha \geq 0.9326$

Table 1: Approximation ratios for QAOA and classical algorithms for different graph types. From [11, 29, 16]

4.2 Classical Algorithms

The proven lower bounds for QAOA are lower than the existing best-known classical algorithms, such as the Goemans-Williamson algorithm. However, the benefit of QAOA comes with its increasing approximation ratio with increasing layers, p, and the fact that this does not depend on the system size. It is difficult to theoretically prove a lower bound for high p due to the increasing complexity, so classical algorithms still give higher theoretical worst-case bounds for approximation ratios.

4.2.1 Goemans-Williamson Algorithm

The Goemans and Williamson (GW) algorithm is the best approximation algorithm for MaxCut problems. They found a 0.87856 approximation ratio, an improvement over the previous best of 0.5 by Sahni and Gonzales [14, 24]. This algorithm is considered a semidefinite program, a type of problem concerned with optimizing a linear cost function with the requirement that the matrix is positive semidefinite [14].

The idea behind the algorithm is to relax the constraint that each vertex can only be two values and instead let it be any vector with a magnitude of 1 [14]. When applying the dot product: $x_i \cdot x_j$, we have a positive semidefinite matrix. Out of the possible solution vectors x_i, choose a random hyperplane through the origin and assign all those vectors that lay above to be part of one partition, and all those below to the other. This procedure is often known as "rounding" in semidefinite programming.

Given this approximation ratio of 0.87856, we can see that the ratios from QAOA do not beat this, but there is still a lot to be studied and tested at larger system sizes and QAOA layers.

4.2.2 Reduced Cubic Graphs

This work improves upon the algorithm created by Feige et al. [13] and is another case of a semidefinite program [16]. The algorithm is described as follows: relax the cost function in order to solve for the solutions, round the solution using a random hyperplane, and perform a local improvement step [16]. It is similar, but uses slightly different techniques than the GW algorithm.

Using this algorithm, Halperin et al. found an approximation ratio of at least 0.9326 for graphs with a degree at most three [16]. This is again higher than the proven lower bounds for QAOA for layers p = 1, 2 and 3.

4.2.3 Comparison to QAOA

The classical algorithms can compute an approximation in polynomial time, which requires significant computational resources for large graphs [14]. In contrast, the computational complexity for QAOA increases with the number of layers, p. The point where QAOA could surpass the classical algorithms depends on the graph structure, number of layers (p) and hardware capabilities. More specific results of QAOA are provided in the next section.

4.3 Comparison to Classic Quantum Algorithms

Recall how phase kickback is a cornerstone to many classic quantum algorithms such as Grover's and Shor's. Phase estimation uses phase kickback to extract the phase of the eigenvalue [5]. The number of qubits we use in this algorithm is the number of decimal places of precision we achieve for the phase. This is similar to QAOA in that with a higher circuit depth (larger number of layers), the accuracy of the approximation improves. However, this also means that in order to achieve high precision in phase estimation, a large number of qubits is required, causing phase estimation to be less suitable on NISQ devices than algorithms like QAOA. QAOA's dependence on the number of layers for accuracy, rather than the number of qubits, combined with its hybrid quantum-classical framework, makes it a strong candidate for execution on NISQ devices.

Grover's is another classic quantum algorithm that provides quadratic speedups for unstructured search problems [5]. Grover's algorithm is able to provide exact solutions, unlike QAOA's approximate solutions. However, Grover's still requires a lot of qubits and minimal errors. Therefore, it will most likely not see practical use until fault-tolerant quantum computers exist. Grover's algorithm and QAOA share a similarity in their iterative approaches within quantum circuits: Grover's algorithm repeatedly applies the Grover operator to amplify the probability of the desired outcome, while QAOA executes multiple layers of its parameterized quantum circuit to optimize the solution.

While these algorithms all have different purposes and applications, they share some similarities in how the circuits are created and in their performance. QAOA acts more as an intermediate algorithm between classical optimization and the long-term potential of fully fault-tolerant algorithms such as Grover's, Shor's, and phase estimation.

4.4 Results

4.4.1 Simulated Results

In 2018, Crooks at Rigetti simulated the performance of QAOA on MaxCut with a variety of layers and graph sizes [8]. He found that at $p \geq 8$, the performance of QAOA exceeds that of the Goemans-Williamson algorithm [8] (see Fig 4).

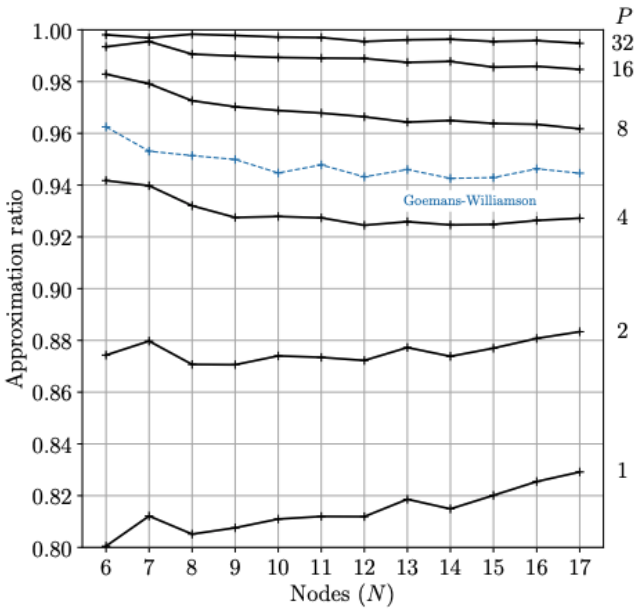

Figure 4: Results from Crooks [8] Approximation ratio of QAOA on MaxCut with various graph sizes (N) and QAOA layers (P). The blue line demonstrates the performance of the Goemans-Williamson algorithm.

Lotshaw et al. later simulated the performance of QAOA on all connected non-isomorphic graphs for $n \leq 9$ vertices [19]. The simulation was done for QAOA layers p = 0, 1, 2, and 3 (see Fig 5). We can see that for p = 2 and 3, QAOA performs better than GW up to 9 vertices. Interestingly, in these experiments, the performance of QAOA seems to converge to a value, suggesting that the performance may stabilize as system size increases.

While no proven theoretical bounds better than classical have been established, QAOA shows immense promise for higher p and larger systems.

4.4.2 Results on Real Quantum Hardware

Running QAOA on real quantum computers introduces many extra factors, including decoherence and noise. While theoretically, as p increases, the approximation ratio should continue increasing for QAOA, this is certainly not the case when we have to consider noise and errors. A

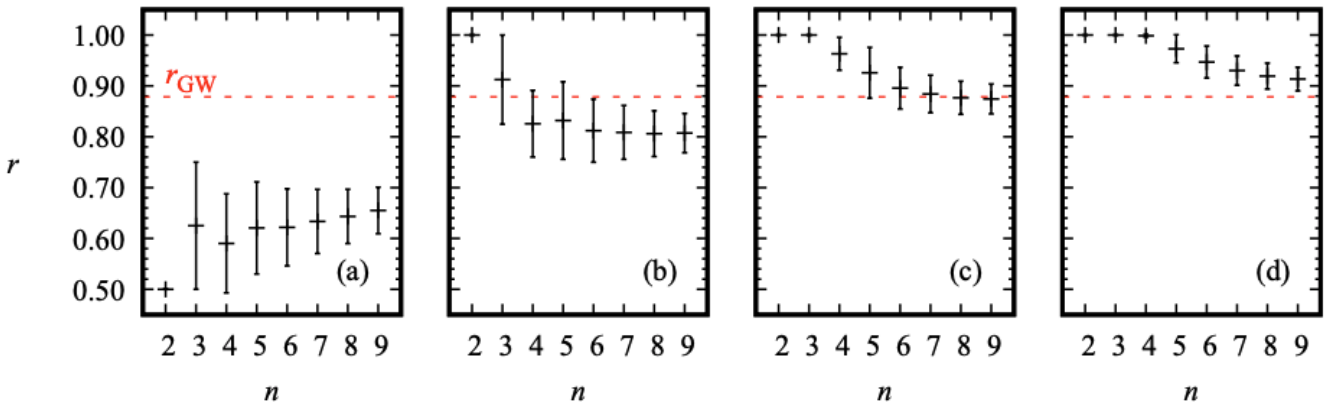

Figure 5: Results from Lotshaw [19] where n is the number of vertices and r is the approximation ratio. The four panels show p = 0, 1, 2, 3 from left to right. The red dashed line shows the approximation ratio of the Goemans-Williamson algorithm.

balance is required between circuit depth and error rates to find the most optimal p.

A team demonstrated the performance of QAOA on the Google Sycamore superconducting processor in 2021 [10]. They ran simulations and experiments on three types of problems: MaxCut on 3-Regular graphs, Hardware grid, and Sherrington-Kirkpatrick (SK) Model. The authors found that performance increased up until p = 3, after which the noise that comes with larger circuit depth outweighed the improvement by increasing p [10].

Bengtsson et al. also performed experiments to test the performance of QAOA on small quantum processors [2]. The authors found that for p = 2 with covariance matrix adaptation evolution strategy for the classical optimization part, the success probability was 96.6% [2]. However, they noticed that the success probability decreased for p = 3 (94.2%) due to increased gate count [2].

An interesting comparison of the performance of QAOA is quantum annealing (QA). The QAOA can be thought of as a form of quantum annealing using discrete time steps and shrinking the time steps to become extremely small ($p \to$ inf) [28]. In this limit, the adiabatic theorem will guarantee that quantum annealing will result in the ground state (see section 2.2.2). The author tested MaxCut on an IBM simulator, IBM Q 16 Melbourne processor, and D-Wave [28]. The performance of the IBM Q processor was very bad compared to the simulator, suggesting that noise has a significant impact [28]. For 2-SAT problems, the D-Qave quantum annealer was able to outperform even the simulated QAOA by a lot [28]. For a problem with 18 vertices, the D-wave machine demonstrated an approximation ratio of 0.91, while QAOA simulated with p = 5 resulted in a ratio of 0.87 [28].

4.5 Variations

Several improvements have been made to QAOA that are worth noting. These include warm-starting QAOA, conditional value at risk, optimizing the circuit, testing other ansatzes and many more. I will briefly discuss these as the scope of the paper does not include an in-depth explanation of QAOA variants.

4.5.1 Warm Starting

The idea behind warm starting quantum optimization (WS-QAOA) is to use classical algorithms to provide the initial state to QAOA, giving it a "headstart" [9]. Any classical algorithm, such as the semidefinite programming techniques in 4.2, can be used to initialize the quantum algorithm. In order to do this, the mixer Hamiltonian needs to be changed to one with the given initial state as ground state [9]. A further variation is to include GW rounding to further optimize the initial state to ensure that WS-QAOA is at least as good as GW [9]. The idea of WS-QAOA is that if we give it an initial state closer to the optimal, it will converge faster than regular QAOA and will, therefore, require lower p.

The authors found that in a simulation, the probability of finding the optimal solution was 5 times higher for WS-QAOA than with standard QAOA for depths $1 \leq p \leq 5$ [9]. They also found that the quality of the solution that the WS-QAOA found was better than standard QAOA.

4.5.2 Recursive QAOA

Bravyi et al. in 2019 propose a non-local version of QAOA, recursive QAOA (RQAOA), addressing some of the limitations caused by the locality and symmetry of QAOA [4]. RQAOA uses QAOA to identify strong correlations

between the variables in the problem at hand [4]. Standard QAOA is first run; then one variable is fixed based on the outcome, which reduces the problem size. This is repeated recursively until the problem is small enough to be solved classically. The authors ran a numerical comparison with p = 1 on system sizes n = 32 and n = 100 to compute the approximation ratios for different problems for standard QAOA and RQAOA [4]. They found that RQAOA performed significantly better than standard QAOA. It is worth noting that while QAOA saw no performance degradation between n = 32 and 100, RQAOA did slightly decrease in performance with n = 100, which is important to keep in mind when considering scaling. Researchers have also applied warm starting to RQAOA (WS-RQAOA), getting even better results than WS-QAOA, RQAOA, and QAOA [9].

4.5.3 CVaR (Conditional Value-at-Risk)

Another improvement to QAOA was made by Barkoutsos et al. in 2020, in which the authors propose that instead of minimizing the expectation value, minimize the CVaR [1]. CVaR is a measure that only takes a fraction of the outcomes, considering only the tail end of the probability distribution [1]. The researchers found that this change allowed the algorithm to find the solution quicker in both simulations and on real quantum hardware [1].

4.6 Limitations

Theoretically, the approximation ratio should increase for QAOA as the number of layers (p) increases. However, this improvement is not always realized due to the noisy and error-prone nature of current quantum hardware. The performance of QAOA can be further affected by increased graph sizes, which require more qubits, amplifying the impact of hardware limitations like decoherence and gate errors. Moreover, optimizing the variational parameters poses a significant challenge; the classical optimization step becomes computationally expensive as the number of parameters grows with larger p. Even though the QAOA is thought to be an algorithm for noisy computers, the current devices still do not match the requirements for the algorithm to outmatch its classical counterparts. Its performance can heavily depend on the structure of the problem.

Sanders et al. optimized circuits of various quantum optimization algorithms, including QAOA, to test if they would be able to be run on a small fault-tolerant quantum computer [25]. They found that despite optimizing the compiled quantum circuits as much as possible, hundreds of thousands of qubits would be necessary to be able to run a problem of size n = 64 with error rates on the order of 10^{-4}. This highlights some of the hardware requirements needed to effectively run QAOA on larger problems.

There is still a lot of research to be done on parameter optimization, ansatz selection, classical optimizers, and circuit optimization [3]. As quantum hardware and classical optimization techniques advance, QAOA will be a good candidate for efficiently approximating combinatorial optimization problems.

5 Conclusion

The Quantum Approximate Optimization Algorithm (QAOA) presents a promising approach to solving combinatorial optimization problems on NISQ devices. By leveraging the power of hybrid quantum-classical methods, QAOA can outperform classical algorithms such as Goemans-Williamson. The approximation ratio of QAOA increases as the number of layers increases rather than system size, which makes it a promising algorithm as the hardware improves. Applications in areas such as logistics, network optimization, and portfolio management highlight its versatility and usefulness.

Many variations and improvements have been made to QAOA, including warm start, recursive, and CVaR, which have reduced the complexity and number of layers required while increasing the approximation ratio. However, there are still challenges regarding the algorithm's performance on current devices compared to classical algorithms. NISQ devices are still too small and noisy to accurately find solutions that are better than the classical algorithms.

There is much future work to be done regarding quantum hardware, error correction and improvements on QAOA to demonstrate its full potential. These developments will bring us closer to achieving quantum advantage and the ability to effectively approximate difficult optimization problems.

References

[1] Panagiotis KI Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. Improving variational quantum optimization using cvar. *arXiv preprint arXiv:1907.04769*, 2019.

[2] Andreas Bengtsson, Pontus Vikstål, Christopher Warren, Maika Svensson, Xiu Gu, Anton Frisk Kockum, Philip Krantz, Christian Križan, Dayoush Shiri, Ida-Maria Svensson, Giovanna Tancredi, Göran Johansson, Per Delsing, Giulia Ferrini, and Jonas Bylander. Improved success probability with greater circuit depth for the quantum approximate optimization algorithm. *arXiv preprint arXiv:1912.10495*, 2019.

[3] Kostas Blekos, Dean Brand, Andrea Ceschini, Chiao-Hui Chou, Rui-Hao Li, Komal Pandya, and Alessandro Summer. A review on quantum approximate optimization algorithm and its variants. *Physics Reports*, 1068:1–66, 2024. A review on Quantum Approximate Optimization Algorithm and its variants.

[4] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to state preparation and variational optimization from symmetry protection. *arXiv preprint arXiv:1910.08980*, 2019.

[5] Kenneth R. Brown. *Introduction to Quantum Engineering*. Brown Lab Publishing, 2024.

[6] M. Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C. Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R. McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, and Patrick J. Coles. Variational quantum algorithms. *Nature Reviews Physics*, 3:625–644, 2021.

[7] Clayton W. Commander. Maximum cut problem (max-cut), 2008.

[8] Gavin E. Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem, 2018.

[9] Daniel J. Egger, Jakub Marecek, and Stefan Woerner. Warm-starting quantum optimization. *arXiv preprint arXiv:2009.10095*, 2020.

[10] Matthew P. Harrigan et al. Quantum approximate optimization of non-planar graph problems on a plana superconducting processor. *arXiv preprint arXiv:2004.04197*, 2020.

[11] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm, 2014.

[12] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

[13] Uriel Feige, Marek Karpinski, and Michael Langberg. Improved approximation of max-cut on graphs of bounded degree. *Journal of Algorithms*, 43(2):201–219, 2002.

[14] Michel Goemans and David Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the Association for Computing Machinery*, 42(6):1115–1145, 1995.

[15] Martin Grötschel and László Lovász. Combinatorial optimization. In R. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of Combinatorics*, chapter 28, pages 1541–1593. Springer, 1995.

[16] Eran Halperin, Dror Livnat, and Uri Zwick. Max cut in cubic graphs. *Journal of Algorithms*, 53(2):169–185, 2004.

[17] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

[18] Grant Kluber. Trotterization in quantum theory, 2023.

[19] Phillip C. Lotshaw, Travis S. Humble, Rebekah Herrman, James Ostrowski, and George Siopsis. Empirical performance bounds for quantum approximate optimization. *Quantum Information Processing*, 20(12), 2021.

[20] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18, 2016.

[21] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5, 2014.

[22] John Preskill. Quantum computing in the nisq era and beyond. *arXiv preprint arXiv:1801.00862*, 2018.

[23] Abraham P. Punnen. *The Quadratic Unconstrained Binary Optimization Problem*. Springer, Cham, Switzerland, 2022.

[24] Sartaj Sahni and Teofilo Gonzales. P-complete problems and approximate solutions. In *15th Annual Symposium on Switching and Automata Theory (swat 1974)*, pages 28–32, 1974.

[25] Yuval R. Sanders, Dominic W. Berry, Pedro C.S. Costa, Louis W. Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush. Compilation of fault-tolerant quantum heuristics for combinatorial optimization. *PRX Quantum*, 1(2), November 2020.

[26] Jules Tilly, Hongxiang Chen, Shuxiang Cao, Dario Picozzi, Kanav Setia, Ying Li, Edward Grant, Leonard Wossnig, Ivan Rungger, George H. Booth, and Jonathan Tennyson. The variational quantum eigensolver: A review of methods and best practices. *Physics Reports*, 986:1–128, 2022.

[27] Eric W. Weisstein. Np-problem.

[28] Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19(7), June 2020.

[29] Jonathan Wurtz and Peter Love. Maxcut quantum approximate optimization algorithm performance guarantees for ¡mml:math xmlns:mml="http://www.w3.org/1998/math/mathml"¿¡mml:mrow¿¡mml:mi¿p¡/mml:mi¿¡mml:mo¿gt;¡/mml:
Physical Review A, 103(4), April 2021.

[30] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Phys. Rev. X*, 10:021067, 2020.