Applying a Quantum Approximate Optimization Algorithm and Grover's Algorithm to an Unweighted Max-Cut Problem

Aneesh Sreedhara
asreedh@ncsu.edu
North Carolina State University
Raleigh, North Carolina, USA

Abstract

We apply a Quantum Approximate Optimization Algorithm (QAOA) and Grover's Algorithm to an unweighted Max-Cut problem with an input graph $G = (V, E)$ with $|V| = 6$ vertices and $|E| = 10$ edges. We develop a QAOA to solve the unweighted Max-Cut problem for G utilizing a Simultaneous Perturbation Stochastic Approximation (SPSA) classical optimizer. Given a number of cuts x we utilize Grover's Algorithm to find the partitioning which yields x cuts. We test both algorithms in ideal and noisy environments and discuss our findings.

Keywords: Unweighted Max-Cut, Variational Quantum Algorithm (VQA), Quantum Approximate Optimization Algorithm (QAOA), Grover's Algorithm

1 Introduction

We aim to apply a Quantum Approximate Optimization Algorithm (QAOA) and Grover's Algorithm to an unweighted Max-Cut problem with an input graph $G = (V, E)$ with $|V| = 6$ vertices and $|E| = 10$ edges and test our algorithms in ideal and noisy environments. We aim to utilize QAOA to solve an unweighted Max-Cut problem for G, and we aim to utilize Grover's Algorithm to find the partitioning of vertices which yields some number of cuts x which we provide.

2 Background

2.1 Unweighted Max-Cut

Max-Cut is a combinatorial optimization problem that is difficult to solve efficiently with classical approaches and is a well-known example of a NP-Hard problem [3]. However, advancements in the field of quantum computing have revealed a number of possibilities with regard to achieving computational speedup over purely classical approaches in solving combinatorial optimization problems such as Max-Cut [2].

In the unweighted Max-Cut problem (which we will refer to as Max-Cut) we are provided an undirected, unweighted graph $G = (V, E)$ and are tasked with partitioning V into disjoint subsets V_1, V_2 such that the number of edges $(i, j) \in E$ where $\text{set}(i) \neq \text{set}(j)$ is maximized [3]. We may define partitions with set $Y = \{y_i\}$ where $y_i = -1$ if vertex $i \in V_1$ and $y_i = 1$ if vertex $i \in V_2$. We may then develop an objective function which we wish to maximize as shown [3].

$$\max \frac{1}{2} \sum_{(i,j)\in E} (1 - y_i y_j)$$

Note that for all $(i, j) \in E$, $\text{set}(i) = \text{set}(j) \implies 1 - y_i y_j = 0$ and $\text{set}(i) \neq \text{set}(j) \implies 1 - y_i y_j = 2$, meaning that for any given term in the summation, if the corresponding edge (i, j) is cut, that term evaluates to 2, and evaluates to 0 otherwise. Then, the result of the summation will be the number of edges which are cut multiplied by 2. We then divide by 2 to obtain the number of edges which have been cut.

There additionally exists the weighted Max-Cut problem for graphs with a set of weights $W = \{w_{ij}\}$ where w_{ij} is the weight of edge (i, j) for which the following objective function that we wish to maximize applies [3].

$$\max \frac{1}{2} \sum_{(i,j)\in E} w_{ij}(1 - y_i y_j)$$

We include this definition to avoid confusion between our interpretation of the Max-Cut problem (unweighted). and the weighted variant. We will focus on the former.

2.2 QAOA

Variational Quantum Algorithms (VQAs) are the leading approach towards achieving quantum advantage with modern Noisy Intermediate-Scale Quantum (NISQ) hardware [2]. To construct a VQA, we first develop a cost function C which we aim to minimize [2]. After this, we develop a quantum operation which depends on a set of parameters θ, and we refer to this operation as the ansatz [2]. We then utilize a classical optimizer to iteratively optimize θ so as to approximate the set of parameters θ^* that minimizes C as shown [2].

$$\theta^* = \arg\min C(\theta)$$

VQAs consist of parameterized quantum circuits to be run on quantum hardware with parameter optimization left to a classical optimizer [2]. Such an approach allows us to limit the depth of quantum circuits, reducing the impact of noise, hence the algorithm's compatibility with current NISQ hardware [2].

One such VQA, the Quantum Approximate Optimization Algorithm (QAOA), is an algorithm developed with the purpose of approximating solutions for combinatorial optimization problems [4]. To construct a QAOA, we begin by encoding the problem into a cost Hamiltonian H_C. We then define a mixer Hamiltonian H_B which is frequently defined as the sum of single-qubit Pauli-X operators for each qubit [4]. We then select initial state $|s\rangle$ to be the uniform superposition of our basis states as follows [4].

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle$$

We then define the following unitary operators dependent on angles γ and β, where $C = H_C$, $B = H_B$, and m and n are the number of clauses and bits in the original combinatorial optimization problem respectively [4].

$$U(C, \gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}$$

$$U(B, \beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta \sigma_j^x}$$

Then, for some $p \geq 1$, and angles $\gamma \equiv \gamma_1 ... \gamma_p$ and $\beta \equiv \beta_1 ... \beta_p$, we define the following state dependent on γ and β [4].

$$|\gamma, \beta\rangle = U(B, \beta_p)U(C, \gamma_p)...U(B, \beta_1)U(C, \gamma_1)\,|s\rangle$$

This represents the ansatz, dependent on parameters $\theta = (\gamma, \beta)$ [4]. The ansatz is a layered circuit which alternates between the problem unitary $U(C, \gamma)$ and the mixer unitary $U(B, \beta)$ a set number of times p [4]. Finally, we utilize a classical optimizer to iteratively optimize θ so as to approximate the θ^* which minimizes the expectation value of H_C.

The selection of an efficient and reliable classical optimizer is critical to the success of VQAs [2]. Gradient descent is a commonly utilized approach for finding the minimum values of a given function via analysis of the function's gradient and is used in a variety of fields including machine learning and quantum computing; it can be thought of as iteratively moving "down" the "slope" of a function in such way that maximizes the rate of descent until a minimum is reached [9]. Given a model $\theta \in \mathbb{R}^d$, loss function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$, learning rate α, and iteration count t, the update rule for gradient descent would look like somewhat like the following [9].

$$\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla \mathcal{L}(\theta^{(t)})$$

The quantum landscape presents a unique set of challenges for implementing such an algorithm, one such challenge being that of performing the numerous, precise measurements required to evaluate the gradient $\nabla \mathcal{L}$ [8, 9]. This has given rise to a number of alternative gradient descent algorithms which are significantly more efficient in this respect, one such alternative being Simultaneous Perturbation Stochastic Approximation (SPSA) [8]. The update rule for SPSA would look like the following [8].

$$\theta^{(t+1)} = \theta^{(t)} - \alpha g^{(t)}(\theta^{(t)})$$

Note that the gradient $\nabla \mathcal{L}$ is replaced with an approximation of the gradient $g^{(t)}$ computed in the following manner [8].

$$g^{(t)}(\theta^{(t)}) = \frac{y_t}{c_t} \begin{bmatrix} \Delta_{t1}^{-1} \\ \Delta_{t2}^{-1} \\ \vdots \\ \Delta_{tp}^{-1} \end{bmatrix}$$

That is to say, the approximation of the gradient is computed by evaluating changes in output resulting from small, random perturbations of its input parameters [8]. SPSA offers a significant speedup over traditional gradient descent due to the replacement of $\nabla \mathcal{L}$ by $g^{(t)}$, as SPSA requires only two function evaluations per iteration to approximate the gradient [8]. Speedup aside, SPSA has been shown to be resilient to noise in the objective function, which is a crucial consideration given the current prevalence of NISQ hardware [8]. Additionally, the noise introduced by SPSA via the random perturbations for gradient approximation help to avoid entrapment in local minima [8]. In general, SPSA has been shown to perform well as an optimizer for VQAs.

2.3 Grover's Algorithm

Grover's algorithm is a quantum algorithm designed with the purpose of performing a generic database search over a set S of $N = 2^n$ records defined such that $\forall s \in S : s \in \{0, 1\}^n$ with function $f : \{0, 1\}^n \to \{0, 1\}$ where $f(s_x) = 1$ for every s_x in the set of solutions S_x and $f(s_x) = 0$ otherwise [5, 6].

Grover's algorithm relies on a gate G which we refer to as the Grover iterate, which in turn relies on an oracle gate O and a diffusion gate D [6]. We encode f into a black box unitary oracle O such that O flips the phase of states which correspond to a solution state as shown [6].

$$O : |s\rangle \mapsto (-1)^{f(s)}$$

We then define a diffusion operator D which flips all amplitudes about the mean as shown [6].

$$U_{0\perp} : \begin{cases} |x\rangle \to -|x\rangle, & x \neq 0 \\ |0\rangle \to |0\rangle \end{cases}$$

$$D = HU_{0\perp}H$$

We then combine O and D to form the Grover iterate as shown [6]. The continued application of G results in the amplification of states which fit our condition f [6].

$$G = OD$$

Following the construction of G, we begin by setting the qubits which represent our solution space into an equal superposition with the use of H gates [6]. We then apply the Grover iterate $\lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ times [6]. After this, we measure the resulting state [6].

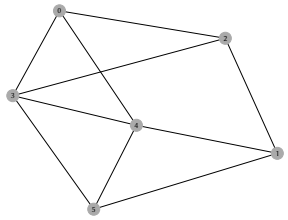

Figure 1. Input Graph $G = (V, E)$ with $|V| = 6$ Vertices and $|E| = 10$ Edges

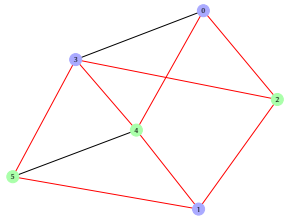

Figure 2. Max-Cut of Input Graph G

3 Problem

Given a graph $G = (V, E)$ with $|V| = 6$ vertices and $|E| = 10$ edges, we aim to utilize QAOA to solve an unweighted Max-Cut problem for G, and we aim to utilize Grover's Algorithm to find the partitioning of vertices which yields some number of cuts x which we provide.

3.1 Graph Definition

We define an undirected, unweighted graph $G = (V, E)$ where $|V| = 6$ and $|E| = 10$. $V = \{0 \rightarrow 5\}$ and $E = \{(0, 2), (0, 3), (0, 4), (1, 2), (1, 4), (1, 5), (2, 3), (3, 4), (3, 5), (4, 5)\}$. We provide an illustration of G in Figure 1.

The Max-Cut of this graph is achieved via the partitioning represented by $Y = \{-1, -1, 1, -1, 1, 1\}$. We provide an illustration in Figure 2, where green nodes are in V_1, blue nodes are in V_2, and red edges represent cut edges. The Max-Cut yields 8 cuts.

Our limitations on G, that is, $|V| = 6$ and $|E| = 10$, result in a fair balance between complexity and practicality for the purpose of demonstrating the application of quantum technologies for solving a Max-Cut problem. Our restrictions permit a sufficient problem size and density so as to provide an effective demonstration of the involved algorithms while keeping the problem easy to interpret for our audiences.

3.2 Algorithm Selection

3.2.1 QAOA. VQAs are the leading approach towards achieving quantum advantage on NISQ hardware and QAOA is a VQA designed specifically for the purpose of solving combinatorial optimization problems such as Max-Cut, which makes QAOA a good choice of algorithm for our purposes [2, 4].

3.2.2 Grover's Algorithm. Grover's algorithm is suitable for the purpose of finding a partition which yields x cut edges for a number of reasons. If we are able to encode a search function into an oracle O, Grover's algorithm presents an efficient solution for identifying states which match the search function. As we will see shortly, we can easily encode the search function for states which yield x cuts into an oracle O. Given this, we may apply Grover's algorithm for our purposes.

4 Method

We provide our methodology and approach for developing both algorithms and detail the environments on which our simulations were run. We utilized Ref [7] for clarification and exploration of involved technologies during our implementation.

4.1 Environment

We utilize the Aer simulator for Qiskit to simulate the execution of our circuits in noisy and ideal environments [1]. For an ideal environment we utilize Aer's default ideal simulator. For a noisy environment we utilize Aer with Qiskit's "FakeCairoV2" backend [1]. Simulations were run on an AMD Ryzen 7 5800X 8-Core Processor (3.80 GHz).

4.2 QAOA

Throughout this section we provide example illustration of portions of a circuit for a graph G_s where $|V_s| = 5, |E_s| = 4, E_s = \{(0, 1), (0, 2), (0, 3), (0, 4)\}$. We do so to provide a simple, conceptual understanding of our approach.

4.2.1 Cost Hamiltonian. We are required to encode our problem into a cost Hamiltonian H_C [2]. Recall that for Max-Cut we aim to find the partitioning Y which satisfies the following expression [3].

$$\max \frac{1}{2} \sum_{(i,j)\in E} (1 - y_i y_j)$$

We could alternatively present this as a minimization problem in which we aim to minimize the number of edges $(i, j) \in E$ where $\text{set}(i) = \text{set}(j)$, shown as follows.

$$\min \frac{1}{2} \sum_{(i,j)\in E} (1 + y_i y_j)$$

We could further alter this expression in such a way that the value of the expression itself changes, but the set Y which correlates with the solution remains the same, shown as follows.

$$\min \sum_{(i,j)\in E} w_{ij} y_i y_j$$

We result with an expression close to being in the form of a Quadratic Unconstrained Binary Optimization (QUBO)

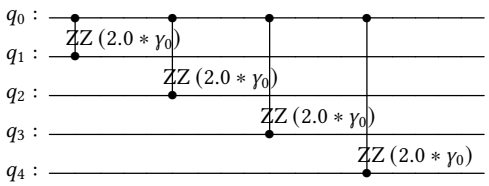

Figure 3. Cost Unitary Operator for G_s

problem with the exception that variables $y_* \in \{-1, 1\}$ rather than $\{0, 1\}$. Ref [10] includes a simple conversion of QUBO problems to Ising Hamiltonians [10]. The first step would normally be to rewrite our cost function in terms of variables in the set $\{-1, 1\}$, but given $y_* \in \{-1, 1\}$, we may omit this step [10]. We continue with the original process to obtain the following cost Hamiltonian H_C [10].

$$H_C = \sum_{(i,j)\in E} w_{ij}\sigma_i^z\sigma_j^z$$

4.2.2 Mixer Hamiltonian. Following from our earlier definitions, we define the mixer Hamiltonian H_B as the sum of single-qubit Pauli-X operators for each qubit as follows [4].

$$H_B = \sum_{j=1}^{n} \sigma_j^x$$

4.2.3 Ansatz. We continue from our earlier definitions. Our initial state $|s\rangle$ is as follows [4].

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_z |z\rangle$$

Our cost unitary $U(C, \gamma)$ is as follows [4].

$$U(C, \gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_\alpha}$$

An illustration of cost unitary $U(C, \gamma_0)$ for G_s is provided in Figure 3, and can be decomposed into the illustration provided in Figure 4. Our mixer unitary $U(B, \beta)$ is as follows [4].

$$U(B, \beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta\sigma_j^x}$$

An illustration of a mixer unitary $U(B, \beta_0)$ for G_s is provided in Figure 5. Our ansatz is as follows and is dependent on parameters $\theta = (\gamma, \beta)$ [4].

$$|\gamma, \beta\rangle = U(B, \beta_p)U(C, \gamma_p)...U(B, \beta_1)U(C, \gamma_1)\,|s\rangle$$

4.2.4 Classical Optimizer. We utilize SPSA to optimize θ so as to minimize the expectation value of H_C.

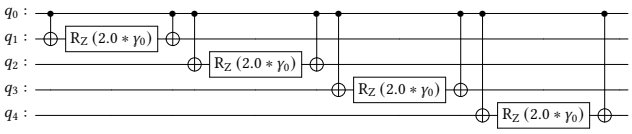

Figure 4. Decomposed Cost Unitary Operator for G_s

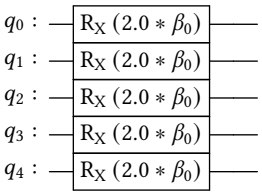

Figure 5. Mixer Unitary Operator for G_s

4.2.5 Circuit. Returning to our actual graph G with $|V| = 6$ and $|E| = 10$, Figure 6 provides a visualization of our quantum circuit. Again, note that this figure represents a circuit for our original graph G, not the simplified variant G_s we defined for conceptual and visual purposes earlier.

4.3 Grover's Algorithm

Throughout this section we provide example illustrations of portions of a circuit for a graph G_s where $|V_s| = 5, |E_s| = 4, E_s = \{(0, 1), (0, 2), (0, 3), (0, 4)\}$. We do so to provide a simple, conceptual understanding of our approach.

4.3.1 Oracle Gate. We must define an oracle gate O which flips the phase of states which fit our condition C_x. For any given state S_i and the corresponding Max-Cut partitioning Y_i it represents, $C_x(S_i)$ should return 1 if the resulting number of cut edges is equal to x. We describe the construction of O as being composed of a number of stages.

Edge Processing. For each edge $(i, j) \in E$ we define an ancilla qubit q_{ij} initialized to $|0\rangle$. We then apply a Controlled-NOT (X_C) gate for control q_i and target q_{ij} and then another X_C for control q_j and target q_{ij}. In result, every $(i, j) \in E$ is represented with an ancilla q_{ij} such that $\text{set}(i) = \text{set}(j) \implies q_{ij} = |0\rangle$ and $\text{set}(i) \neq \text{set}(j) \implies q_{ij} = |1\rangle$. This process is illustrated for G_s in Figure 7.

Cut Summing. We define $\lceil \log_2(|E| + 1) \rceil$ ancilla qubits to hold the sum of cut edges and initialize each qubit to $|0\rangle$. We then increment the value stored in these qubits by 1 for each edge ancilla q_{ij} where $q_{ij} = |1\rangle$. This is accomplished with the use of Multi-Controlled-NOT (MCX). To perform an increment, logically, for each sum qubit, going in descending order of significance, we flip it if all less-significant bits are $|1\rangle$. This process is illustrated for G_s in Figure 8.

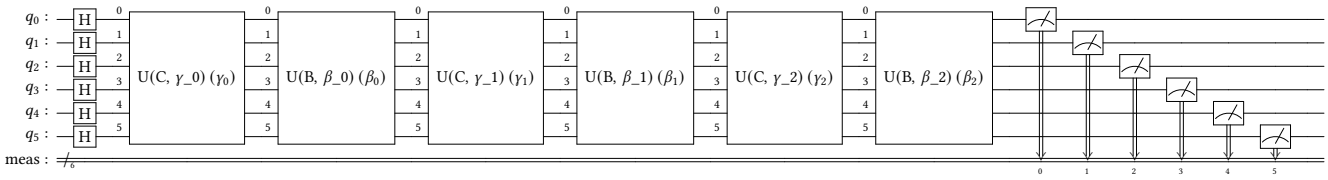

Figure 6. Circuit for QAOA for Max-Cut for G

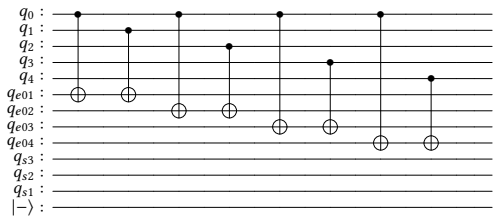

Figure 7. Oracle Edge Processing Step for G_s

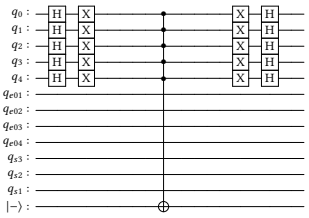

Figure 10. Diffusion Operator for G_s

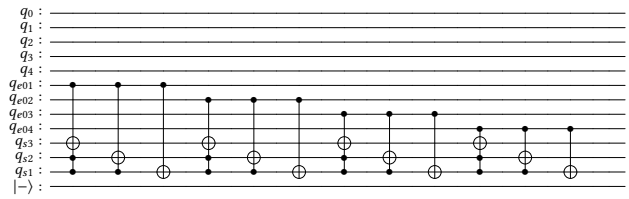

Figure 8. Oracle Cut Summing Step for G_s

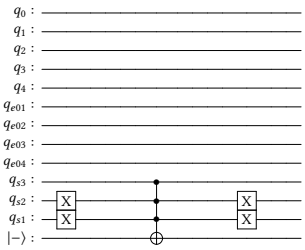

Figure 9. Oracle Phase Flip Step for G_s for $x = 4$

Oracle Phase Flip. We flip the phase of states which meet our condition by flipping the phase of our summation ancillas should they represent our target value. These changes cascade back to our initial qubits during cleanup as we reverse the changes we have made up until this point. Note that we put an ancilla qubit into the $|-\rangle$ state to facilitate phase flips. We provide an illustration for G_s in Figure 9.

Cleanup. We simply do all the operations leading up to the oracle phase flip in reverse. While doing so, our phase flip, if one occurred, propagates.

4.3.2 Diffusion Operator. Our implementation of D is fairly standard and uses the same ancilla utilized by O for phase flipping. We provide an illustration for G_s in Figure 10.

4.3.3 Circuit. Returning to our actual graph G with $|V| = 6$ and $|E| = 10$, Figure 11 provides a visualization of our quantum circuit. Again, note that this figure represents a circuit for our original graph G, not the simplified variant for G_s we defined for conceptual and visual purposes earlier. Note that we set the ancilla used by both O and D to $|-\rangle$ at the start and reverse this at the end.

5 Results

5.1 QAOA

5.1.1 Ideal Simulator. Figures 12 and 13 illustrate the SPSA optimization convergence and the output distribution respectively after running QAOA with SPSA for G for $p = 3$ and and SPSA Iterations $= 300$ on an ideal simulator. As demonstrated, the Max-Cut partition is correctly determined. State $|001011\rangle$ corresponds to $Y = \{-1, -1, 1, -1, 1, 1\}$ and state $|110100\rangle$ is an equivalent, correct answer.

5.1.2 Noisy Simulator. Figures 14 and 15 illustrate the SPSA optimization convergence and the output distribution respectively after running QAOA with SPSA for G for $p = 3$ and and SPSA Iterations $= 300$ on a noisy simulator. As demonstrated, the Max-Cut partition is correctly determined. State $|001011\rangle$ corresponds to $Y = \{-1, -1, 1, -1, 1, 1\}$ and state $|110100\rangle$ is an equivalent, correct answer.

5.2 Grover's Algorithm

5.2.1 Ideal Simulator. Figure 16 illustrates the output distribution after running Grover's Algorithm for G for $x = 8$ and and Grover Iterations $= 6$ on an ideal simulator. As demonstrated, the partition which results in $x = 8$ cuts - that is, the Max-Cut partition - is correctly identified. State

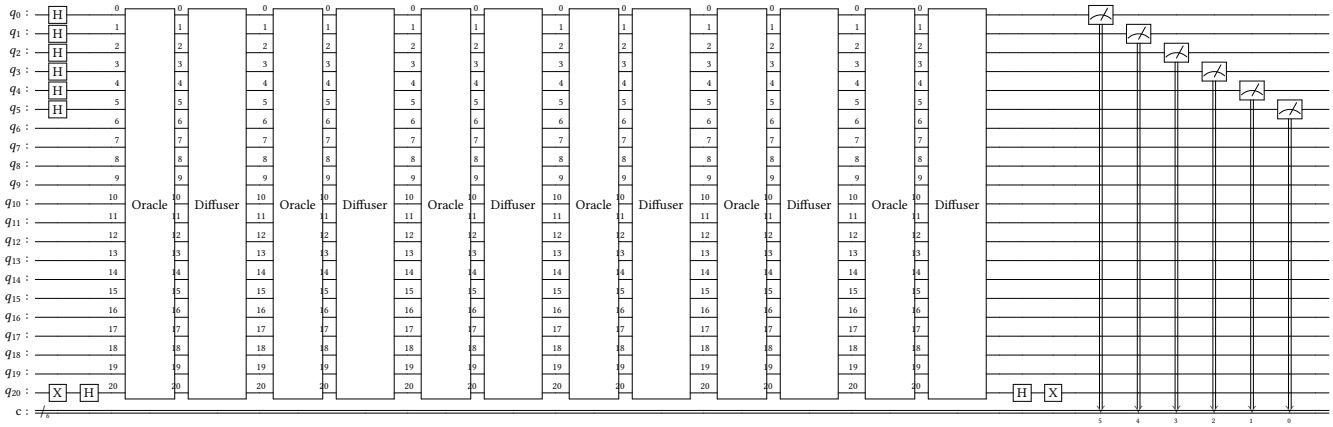

Figure 11. Circuit for Grover's Algorithm for Max-Cut for G

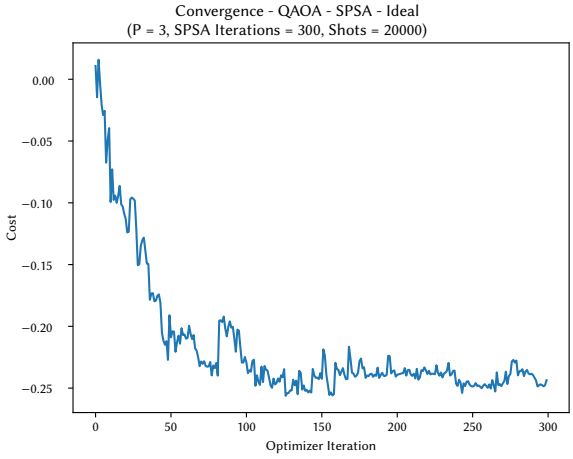

Figure 12. Optimizer Convergence for QAOA with SPSA on Ideal Simulator for $P = 3$, SPSA Iterations = 300

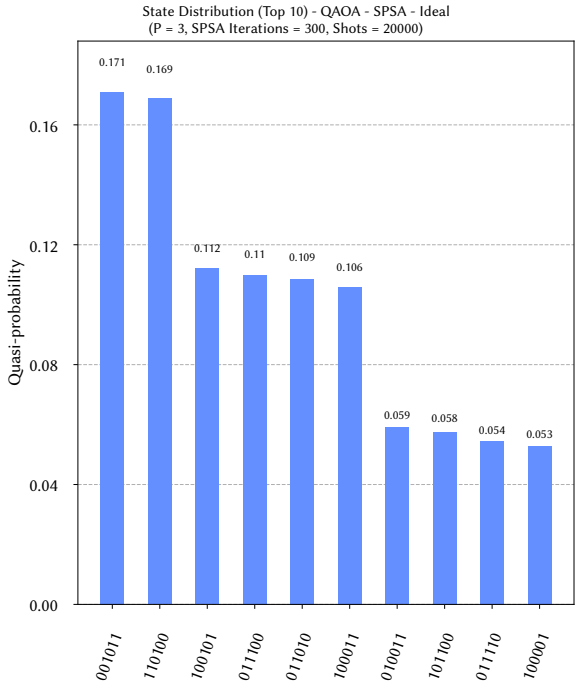

Figure 13. Top 10 Quantum State Distribution for QAOA with SPSA on Ideal Simulator for $P = 3$, SPSA Iterations = 300

$|001011\rangle$ corresponds to $Y = \{-1, -1, 1, -1, 1, 1\}$ and state $|110100\rangle$ is an equivalent, correct answer.

5.2.2 Noisy Simulator. Figure 17 illustrates the output distribution after running Grover's Algorithm for G for $x = 8$ and and Grover Iterations $= 6$ on a noisy simulator. As demonstrated, the partition which results in $x = 8$ cuts - that is, the Max-Cut partition - is *not* correctly identified.

6 Discussion

6.1 QAOA

6.1.1 Ideal Simulator. Runs of QAOA with SPSA on an ideal simulator provide important insights on the convergence and accuracy of our algorithm in an ideal environment.

Figure 12 demonstrates a rapid approach towards a local minimum for parameters θ within the initial iterations followed by a more gradual approach towards a global minimum. The cost function does not stabilize entirely, but this is to be expected, due to the stochasticity introduced by SPSA, and is in fact welcome, as this behavior helps prevent entrapment in local minima. Aforementioned fluctuations notwithstanding, we observe relatively smooth convergence, which is also to be expected, as any further fluctuations as a result of noise are not to be expected in an ideal simulator.

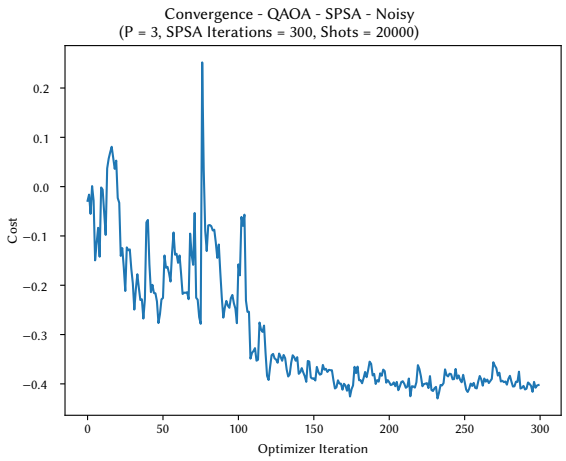

Figure 14. Optimizer Convergence for QAOA with SPSA on Noisy Simulator for $P = 3$, SPSA Iterations = 300

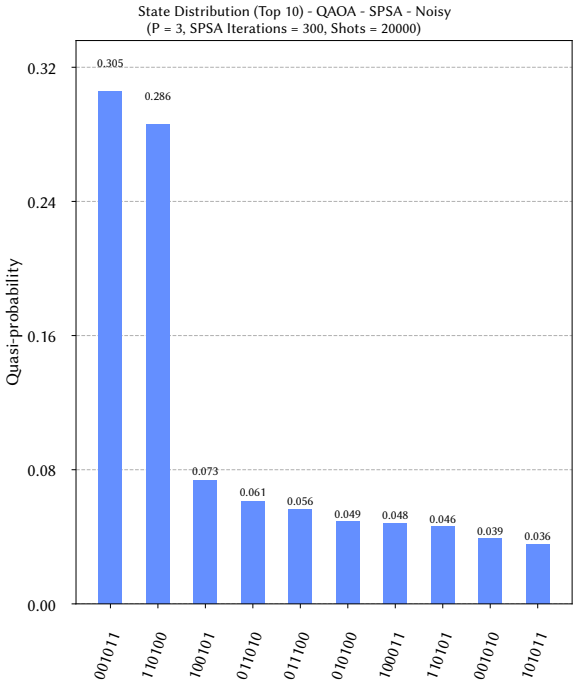

Figure 15. Top 10 Quantum State Distribution for QAOA with SPSA on Noisy Simulator for $P = 3$, SPSA Iterations = 300

While Figure 13 demonstrates highest probabilities for solution states, it also demonstrates middling probabilities for non-solution states. This is to be expected, as while the algorithm attempts to find a global minimum, it may still struggle to escape local minima. So long as the probabilities for solution states remain consistently higher than those

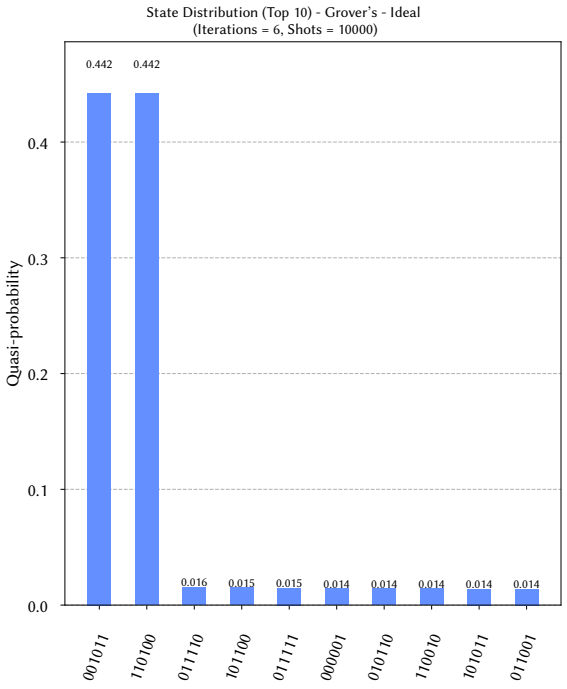

Figure 16. Top 10 Quantum State Distribution for Grover's Algorithm on Ideal Simulator

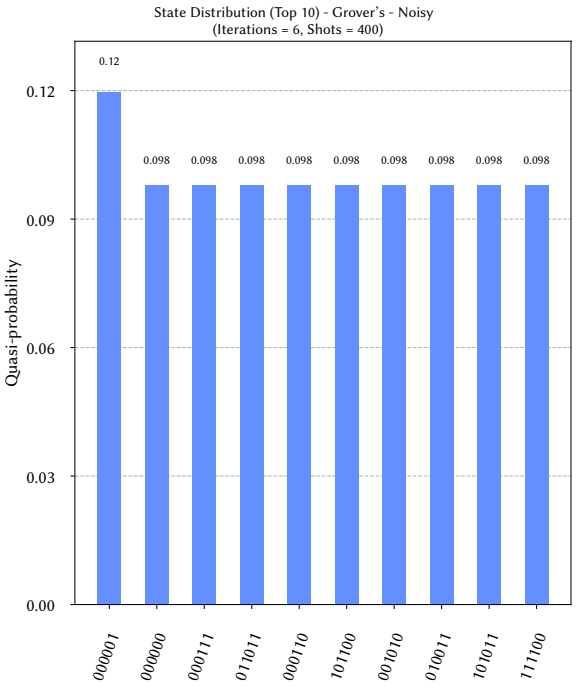

Figure 17. Top 10 Quantum State Distribution for Grover's Algorithm on Noisy Simulator

of states which provide sub-optimal solutions, this is not a cause for concern.

6.1.2 Noisy Simulator. Runs of QAOA with SPSA on a noisy simulator provide important insights on the convergence and accuracy of our algorithm in a noisy environment.

Figure 14 demonstrates a level of instability during the optimization process, which is to be expected, due to noise introduced in the results of the objective function due to the noisy environment. This is entirely predictable, and very much contributed to our reasoning in choosing SPSA, an optimizer known to perform well with noisy objective functions, as our optimizer. Following this, despite the noise, the optimizer ultimately converges on a minimum, with continued fluctuations resulting from noise introduced by both the environment and SPSA helping to prevent entrapment in a local minima.

While Figure 15 demonstrates high probabilities for solution states and low probabilities for non-solution states, it is important that we note that this behavior is not entirely consistent. The distributions outputted by runs of our algorithm in the noisy simulator varied widely in this regard. While the correct solution was obtained the majority of the time, suboptimal solutions were obtained in a number of instances.

6.1.3 Comparison. Comparing runs in noisy and ideal environments yields a number of observations important to understanding the performance of our algorithms.

Predictably, convergence proved to be significantly smoother in an ideal environment as compared to a noisy environment, given that the optimizer did not have to deal with external noise introduced by the environment. While convergence was more volatile in a noisy environment, SPSA demonstrated its resilience to noise in the objective function, eventually converging on a minimum. Given that environmental noise was simulated based on real quantum hardware, this demonstrates viability for use with NISQ hardware [1].

A view of the probability distributions demonstrate some interesting variations. For the ideal simulation, probabilities are more distributed across sub-optimal states, which is a favorable result, as having the ability to find sub-optimal solutions, even if the optimal solution in some cases is not directly found, is valuable in its own right. Meanwhile, distributions for the noisy environment are inconsistent in nature. While the distribution we provide heavily favors the optimal solution, this behavior is inconsistent, and there are many instances in which the distribution appears significantly more distributed. This is likely a result of noise introduced by the environment, in combination with stochasticity introduced by other portions of the algorithm's implementation.

6.2 Grover's Algorithm

6.2.1 Ideal Simulator. Runs of Grover's Algorithm in an ideal environment provide insight into the theoretical capability of our implementation.

Figure 16 demonstrates the effectiveness of our implementation in an ideal environment, correctly identifying the two states which represent the solution partitioning. Such a result is consistent throughout runs, which is to be expected, given that the algorithm is run in an ideal environment.

6.2.2 Noisy Simulator. Figure 14 demonstrates that our implementation does not fair well on a noisy simulator. This is entirely predictable given the size of the circuit. The size and depth of the circuit are simply to high to be run effectively in noisy environments.

7 Conclusion

We applied QAOA and Grover's Algorithm to an unweighted Max-Cut problem with an input graph $G = (V, E)$ with $|V| = 6$ vertices and $|E| = 10$ edges and tested our algorithms in both ideal and noisy environments. With both algorithms, we observed favorable performance in ideal conditions. With QAOA, we found promising results in the noisy environments, which is to be expected, as QAOA, being a VQA, is popular in-part due to its compatibility with modern NISQ hardware. Meanwhile, Grover's Algorithm, also predictably, failed to perform well in a noisy environment, due to the size and depth of the circuit. With regard to QAOA, improvements could be made in several portions of our implementation, including parameter initialization, optimizer tuning, and further depth analysis. Future work may include the exploration of other methods to improve the performance of the algorithm in noisy environments.

References

[1] Amira Abbas, Stina Andersson, Abraham Asfaw, Antonio Corcoles, Luciano Bello, Yael Ben-Haim, Mehdi Bozzo-Rey, Sergey Bravyi, Nicholas Bronn, Lauren Capelluto, Almudena Carrera Vazquez, Jack Ceroni, Richard Chen, Albert Frisch, Jay Gambetta, Shelly Garion, Leron Gil, Salvador De La Puente Gonzalez, Francis Harkins, Takashi Imamichi, Pavan Jayasinha, Hwajung Kang, Amir h. Karamlou, Robert Loredo, David McKay, Alberto Maldonado, Antonio Macaluso, Antonio Mezzacapo, Zlatko Minev, Ramis Movassagh, Giacomo Nannicini, Paul Nation, Anna Phan, Marco Pistoia, Arthur Rattew, Joachim Schaefer, Javad Shabani, John Smolin, John Stenger, Kristan Temme, Madeleine Tod, Ellinor Wanzambi, Stephen Wood, and James Wootton. 2020. Learn Quantum Computation Using Qiskit. https://qiskit.org/textbook/

[2] 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. 2021. Variational quantum algorithms. *Nature Reviews Physics* 3, 9 (Aug. 2021), 625–644. https://doi.org/10.1038/s42254-021-00348-9

[3] Clayton Commander. 2008. Maximum cut problem, MAX-CUT. (01 2008). https://doi.org/10.1007/978-0-387-74759-0_358

[4] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 [quant-ph]

[5] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (Philadelphia, Pennsylvania, USA) *(STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219. https://doi.org/10.1145/237814.237866

[6] Phillip Kaye, Raymond Laflamme, and Michele Mosca. 2006. *An introduction to quantum computing*. OUP Oxford.

[7] OpenAI. 2023. ChatGPT. https://www.openai.com/. Accessed: 2023-12-05.

[8] James C. Spall. 1997. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica* 33, 1 (1997), 109–112. https://doi.org/10.1016/S0005-1098(96)00149-5

[9] Ryan Sweke, Frederik Wilde, Johannes Meyer, Maria Schuld, Paul K Fährmann, Barthélémy Meynard-Piganeau, and Jens Eisert. 2020. Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum* 4 (2020), 314.

[10] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. 2018. Quantum approximate optimization algorithm for MaxCut: A fermionic view. *Physical Review A* 97, 2 (Feb. 2018). https://doi.org/10.1103/physreva.97.022304