ECE 520 Final Project: Quantum Approximate Optimization Algorithm

Kailiang Nan

December 16, 2022

1 Introduction

There is a growing interest in using quantum technology to solve the problems of combinatorial optimization. However, quantum computers are still too noisy to apply large-scale circuits to them. John Preskill described the current stage of quantum device as 'Noisy intermediate-scale quantum'(NISQ) era.[1] We cannot get very precise outcomes by simply running optimization algorithms on quantum computers. That's why hybrid quantum-classical algorithms are designed. Farhi et al.[2] introduced the Quantum Approximate Optimization Algorithm(QAOA) in 2014. This algorithm targets at finding good approximate solutions to the problem of maximizing or minimizing the objective function under the binary constraint.

1.1 Principle

Basically, QAOA is a trotterized quantum adiabatic algorithm. Adiabatic quantum computation takes the advantage of Adiabatic Theorem. It states, if we begin with a ground state $H(0) \rightarrow |\psi(0)\rangle$ of some time-dependent Hamiltonian $H(t)$ where $t \in [0, 1]$ and evolve our quantum system with this Hamiltonian slowly, we end up with the ground state $H(1) \rightarrow |\psi(1)\rangle$ of this Hamiltonian at $t = 1$. The ground state corresponds to the lowest energy of the system, and the same thing is also valid for highest energy state. In the QAOA, we use this adiabatic theorem to find the highest or lowest energy eigenstates of our problem Hamiltonian H_C by initializing our quantum system with the highest/lowest energy eigenstates of some simple Hamiltonian H_M. The Hamiltonian of the system is slowly evolved from the H_M to H_C, which resembles the adiabatic process. In the ideal case, the quantum system is supposed to be in the state that has the highest/lowest energy of our problem Hamiltonian H_C after evolution. We then sample states from it to get the solution of the problem. The formula below describe the time-dependent Hamiltonian $H(t)$

$$H(t) = (1 - t)H_M + tH_C \qquad t \in [0, 1] \tag{1}$$

The time-dependent Schrodinger equation gives us the time evolution of the eigenstates of a certain Hamiltonian H

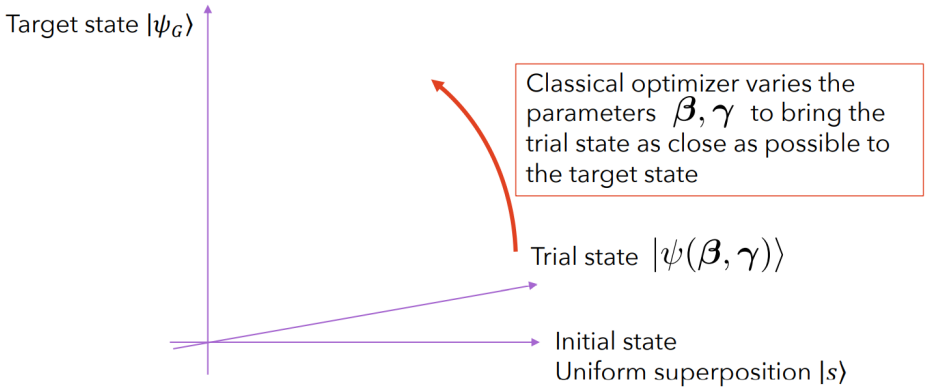
$$|\psi(t)\rangle = e^{-iHt/\hbar} |\psi(0)\rangle \tag{2}$$

Since we want to simulate the process of time evolution of the system, we need to consider how to simulate the process of $e^{-i((1-t)H_M+tH_C)t/\hbar}$ in the quantum circuit. Fortunately, we can use the trotterization process to approximate this evolution. We have learned Trotter-Suzuki formula in the class

$$e^{-i(H_1+H_2)t} \approx (e^{-iH_1t/r}e^{-iH_2t/r})^r \tag{3}$$

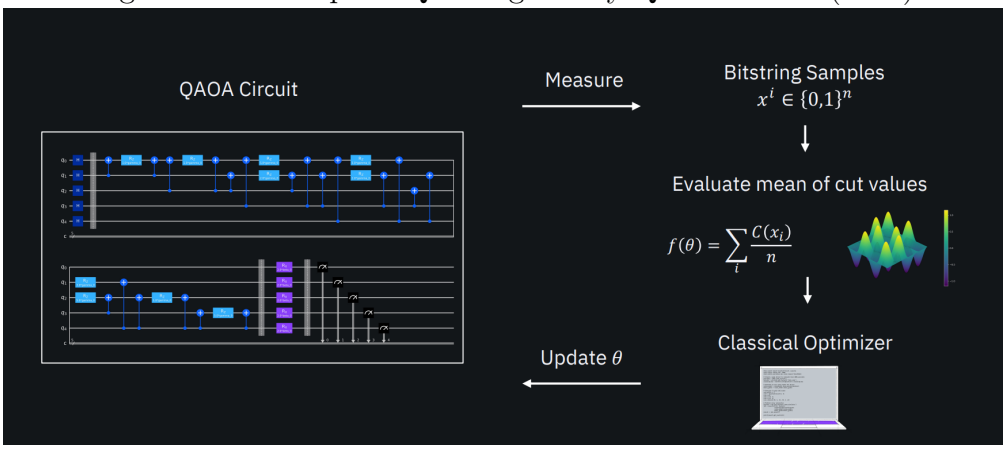
Trotterization means, rather than giving a quantum circuit that corresponds to the real Hamiltonian of the simulated process, we apply the circuits corresponding to H_M and H_C in sequence. According to the Trotter-Suzuki formula, we can get a good approximation of the circuit corresponding to the original Hamiltonian if we apply them multiple times. The precision of approximation can be controlled by changing the repeating time p.
We also need to find some parameters for each layer in our circuit. A single layer is a quantum circuit that mimic the time evolution of either the problem Hamiltonian or the initial state Hamiltonian. These two types of layers are called cost layer and mixer layer respectively. The parameters are used to characterize the time elapse of each system. A series of γ's and β's are usually used in QAOA.
The initial parameters are randomly given, meaning the quantum circuit mimics the Hamil-

Figure 1: Principle of QAOA



tonian at a random time t. The measurements of the circuit give the probability distribution of the states being observed. We calculate the energy at each state and get the average energy of measurement. Then, we update the parameters in classical way. Some classical optimizers, such as ADAM(Adam and AMSGRAD optimizers) and COBYLA(Constrained Optimization By Linear Approximation optimizer), are utilized to update the value of parameters. We repeat the same process for new parameters and do it many times until a good solution is found. The last parameters given by the simulation could be used to characterize the Hamiltonian at $t = 1$. The quantum circuit simulated with this set of parameters is supposed to be in the state that represents the best solution of our defined problem, and the measured energy of this state represents the best objective value we can get.

Figure 2: The steps of QAOA given by Qiskit tutorial(IBM)



1.2 Problem Definition

In this project, the MaxCut problem has been simulated. In the MaxCut problem, the input is a weighted graph. We want to divide the graph vertices into two disjoint sets such that the cumulative weight of edges linking vertices from different sets is maximized.
Assume $G = (V, E)$ is a weighted graph that has n vertices. The MaxCut problem is to assign each vertex either the value 0 or 1. A vector $x \in \{0, 1\}^n$ can be used to describe a strategy of partition which we call *a cut*. The weight of a cut x is determined by the cost function

$$C(x) = \sum_{i,j=1}^{n} W_{ij} x_i (1 - x_j) \tag{4}$$

Note that $W_{ij} = W_{ji}$ is the weight of the edge between vertices v_i and v_j, and each edge weight contributes exactly once to the sum.

Figure 3: The graph simulated in this project

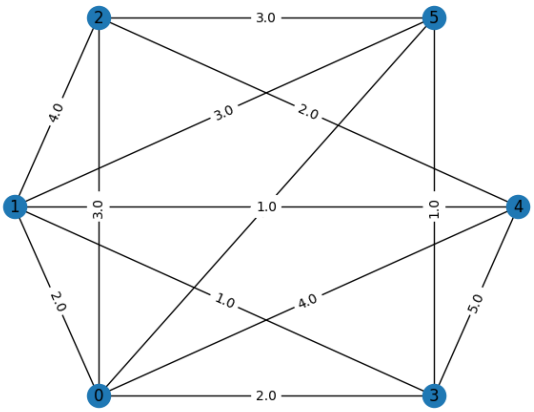
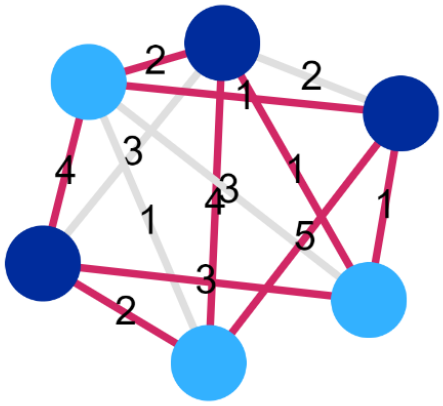


Figure 4: The best solution for the graph



1.2.1 QUBO

To encode the problem into our quantum circuit, we should know that MaxCut problem is a quadratic unconstrained binary optimization(QUBO) problem. The mathematical formulation of the QUBO can be expressed as below.

$$\begin{aligned} \min_{x} \quad & x^T Q x + c^T x \\ \text{s.t.} \quad & x \in \{0,1\}^n \end{aligned} \tag{5}$$

We can reformulate the MaxCut problem in the standard form for QUBO as written above.

$$\begin{aligned} \sum_{i,j=1}^{n} W_{ij} x_i \left(1 - x_j\right) &= \sum_{i,j=1}^{n} W_{ij} x_i - W_{ij} x_i x_j \\ &= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} W_{ij} \right) x_i - \sum_{i,j=1}^{n} W_{ij} x_i x_j \\ &= c^T x + x^T Q x \end{aligned} \tag{6}$$

$$\text{where } Q_{ij} = -W_{ij} \text{ and } c_i = \sum_{j=1}^{n} W_{ij} \text{ and } \quad x \in \{0,1\}^n$$

1.3 Ansatz

The circuit is initialized with a uniform superposition of all possible solutions.

$$|+\rangle = \sum_{x \in \{0,1|^n} \frac{1}{\sqrt{2^n}} |x\rangle \tag{7}$$

The mixer Hamiltonian H_M(The Hamiltonian with an easy highest energy eigenstate) is defined as the sum of single Pauli X-operators on all qubits

$$H_M = \sum_{i=1}^{n} X_i. \tag{8}$$

H_C is the cost Hamiltonian which encodes the objective function of the optimization problem.

$$H_C |x\rangle = \left(x^T Q x + c^T x\right) |x\rangle = \left(\sum_{i,j=1}^{n} x_i Q_{ij} x_j + \sum_{i=1}^{n} c_i x_i \right) |x\rangle \tag{9}$$

Recall the following relations of Pauli-Z operator.

$$Z_i |x\rangle = (-1)^{x_i} |x\rangle = \left(1 - 2x_i\right) |x\rangle \tag{10}$$

This gives us

$$x_i |x\rangle = \frac{I - Z_i}{2} |x\rangle, \tag{11}$$

By plugging equation (11) into equation (9), we get the following Hamiltonian.

$$H_C = \sum_{i,j=1}^{n} \frac{1}{4} Q_{ij} Z_i Z_j - \sum_{i=1}^{n} \frac{1}{2} \left(c_i + \sum_{j=1}^{n} Q_{ij} \right) Z_i + \left(\sum_{i,j=1}^{n} \frac{Q_{ij}}{4} + \sum_{i=1}^{n} \frac{c_i}{2} \right) \tag{12}$$

A variational from involving R_Z and R_{ZZ} gates in the cost layer and R_X gates in the mixer layer is obtained by exponentiating two Hamiltonians.

$$e^{-i\beta H_M} = \prod_{i=1}^{n} R_{X_i}(2\beta) \quad e^{-i\gamma H_C} = \prod_{\substack{i,j=1 \\ i \neq j}}^{n} R_{Z_i Z_j} \left(\frac{1}{2} Q_{ij} \gamma \right) \prod_{i=1}^{n} R_{Z_i} \left(\left(c_i + \sum_{j=1}^{n} Q_{ij} \right) \gamma \right)$$

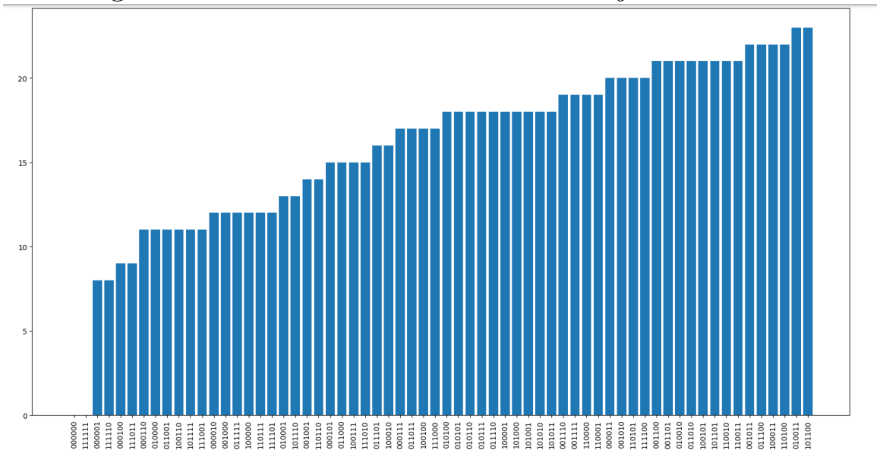
Where β and γ characterize the time elapse of the system. In this project, the repeat time p is set to 1 since the MaxCut problem we defined is not difficult to solve.

2 Experiment Result Analysis

2.1 Get All Possible Solutions

Since the problem defined in this project is relatively easy, the problem is first solved using a brute-force way where all the solutions and their corresponding objective values are found. The bitstring [1,0,1,1,0,0] has the largest objective value. Note the first bit here corresponds

Figure 5: Bar chart of solutions v.s. objective values

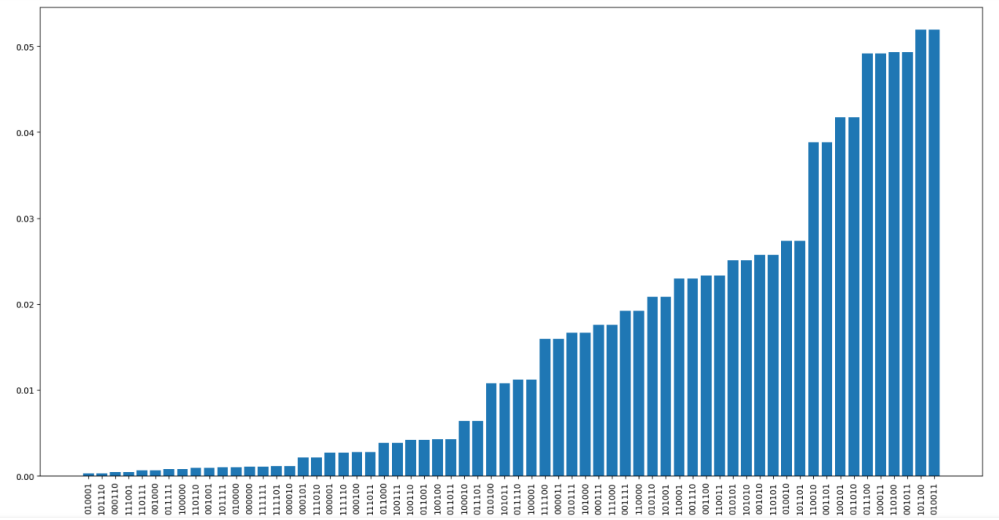


to the first vertex in the graph. For each bitstring, there is a symmetric bitstring where each bit is flipped. The two bitstrings represent the same solution. [1,0,1,1,0,0] and [0,1,0,0,1,1] make no difference for a partition problem. Figure 4 demonstrates the best solution for the defined problem.

2.2 Solution of QAOA

The problem graph shown by Figure 3 was encoded into problem Hamiltonian and then a QAOA circuit was built based on this Hamiltonian. The quantum circuit was then simulated with Qiskit's Aer statevector simulator. Figure 6 shows the result of the simulation. The

Figure 6: Probability Distribution of Simulation



bitstring with highest probability given by QAOA is exactly the answer we found in brute-force way.

2.3 Understand Optimization Process

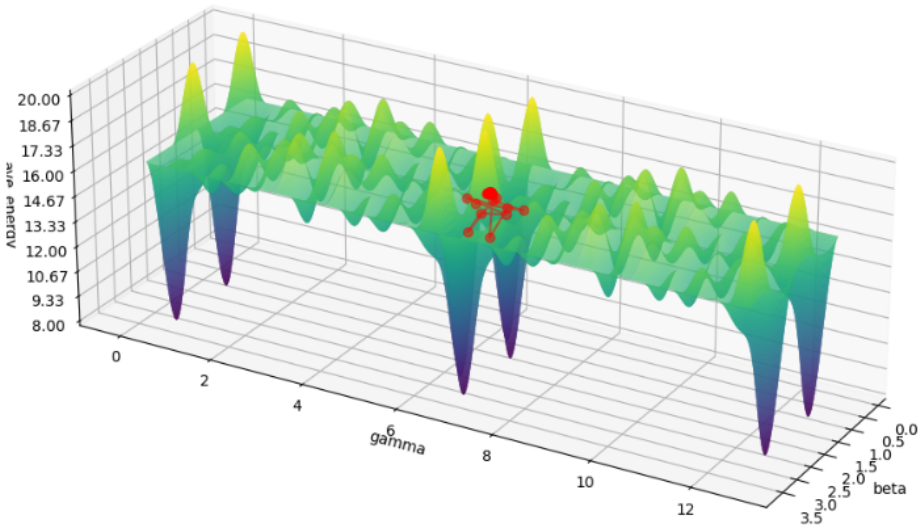
The solution of QAOA is good. However, the simulation was done with Qiskit's internal optimizer and did not show any extra procedure. To understand how the optimization process was going, some callback functions were written to capture the optimization process of the QAOA. To show the process more clearly, a grid search of all the possible values of parameters and their corresponding average eigenvalues is conducted to find the optimization landscape. Figure 7 is the diagram drawn by Python program. The searched parameter is indicated by the red points on the landscape. The red points are connected by the red lines to show the trajectory of the search.

Figure 8 shows the optimization curve during the iterations of finding new parameters. We can see the QAOA finds some higher energies after about 18 iterations and becomes stable after that. However, the best set of parameters found by it is only a local optima. Although it gives correct solutions for the simple problem defined by this project, it may fail if the problem is getting more complicated.

2.4 Add More Layers

In theory, if we increase the depth of QAOA circuit, we are guaranteed to obtain better approximation of the real problem we want to simulate. Thus, the factor of p is explored.

Figure 7: Optimization Landscape and Search Trajectory



Firstly, We look at some theories about this. Let M_p denotes the maximal energy value a QAOA circuit of depth p can obtain.

$$\boldsymbol{M}_p = \max_{\beta,\gamma \in R^p} \langle \psi_p(\beta,\gamma) \left| H_C \right| \psi_p(\beta,\gamma) \rangle \tag{13}$$

The theory says the following relation holds

$$M_{p+1} \geq M_p \tag{14}$$

According to the adiabatic theorem, we are able to reach the maximal cost function value $C_{\max}$ in the infinite limit.

$$\lim_{p\to\infty} M_p = C_{\max}. \tag{15}$$

In this project, the change of average energy, running time and number of iterations with respect to the change of p were explored. Figure 9 shows the result. It is clear we can get a better average energy if we increase p. However, the running time and the number of evaluations also increase the p. This means the better result comes at the cost of longer time and more iteration of parameter explorations. The trade-off between precision and the running cost should be considered carefully before applying QAOA to solve some practical problems which is usually more complicated.

Figure 8: Optimization Curve

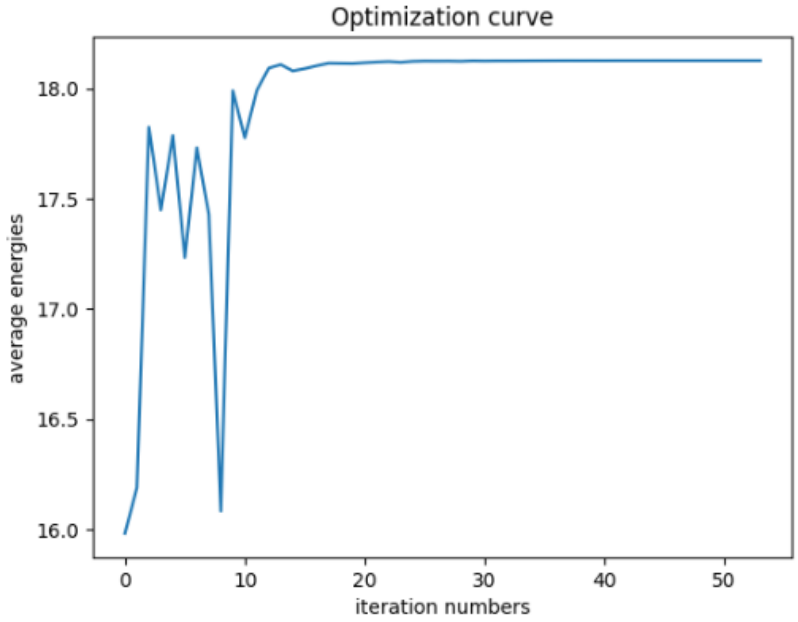
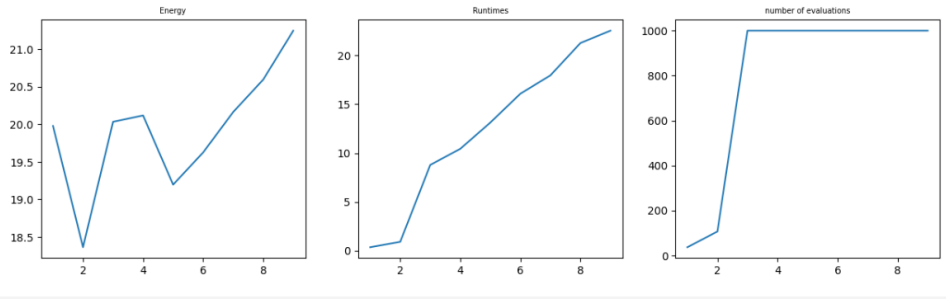


Figure 9: The factor of p



3 Comparison and Progress

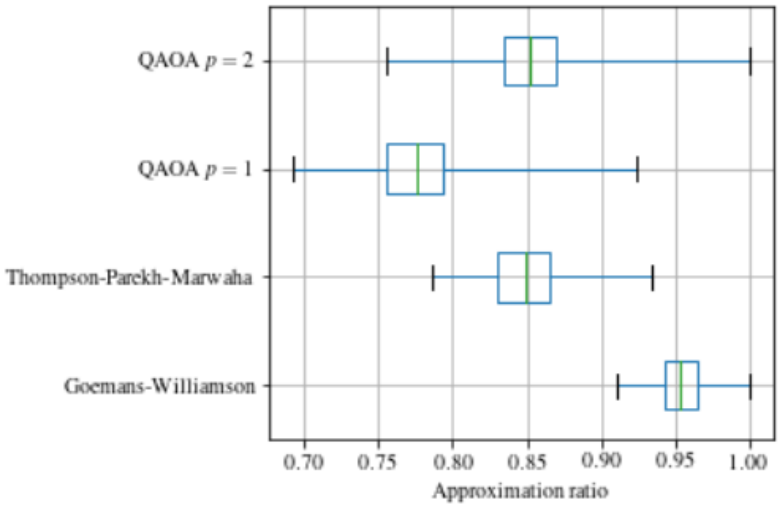
3.1 Comparison with Classical Algorithm

In general, classical methods use relaxation to make the MaxCut problem more easier to solve. Some projection strategies are then applied to get the final answer of the problem. Since this project did not implement the classical algorithms of solving MaxCut problem, this report references the comparison result from Ruslan Shaydulin. Shaydulin design a Python toolkit *QAOAKit* for exploring the behaviors of QAOA.[3] In this toolkit, the comparison between quantum and classical algorithm for solving MaxCut problem is made.

From Figure 10, it is easy to see that, at p=2, QAOA can match the performance of the Thompson-Parekh-Marwaha algorithm. However, it is still worse than Goemans-Williamson algorithm. We know in theory, if p approaches infinity, the QAOA must have a better performance than that of the classical algorithm regardless of the computational cost. However, at

the current stage, error propagates and accumulates if the quantum circuit is too deep. By now, quantum approximate algorithm is still not able to supersede the classical algorithm.

Figure 10: Classical Algorithm v.s QAOA



3.2 Recent Progress on QAOA

Although the practical application of QAOA is still limited by the fidelity of quantum hardware, we could still find some ways to make QAOA behave better. In this project, two adaptations of QAOA algorithm are implemented. The result shows a better performance than that of the original algorithm.

3.2.1 CVaR

In 2020, Barkoutsos et. al. introduced a method called Conditional Value-at-Risk(CVaR) to speed up optimization process.[4] To put it simple, since we want to find the state with maximum measured average energy, we replace the mean of measured cut of all the values obtained from the simulations with a new value which only takes into account a fraction α of the highest measured cut values.

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} c_i \rightarrow f(\theta) = \frac{1}{\lceil \alpha n \rceil}\sum_{i=1}^{\lceil \alpha n \rceil} c_i \tag{16}$$

The experimental result shows CVaR can really give a state with higher average energy after a fixed number of iterations. Since only one experiment is conducted in this project, we cannot rule out the possibility of contingency. More results are needed to prove the superiority of CVaR.

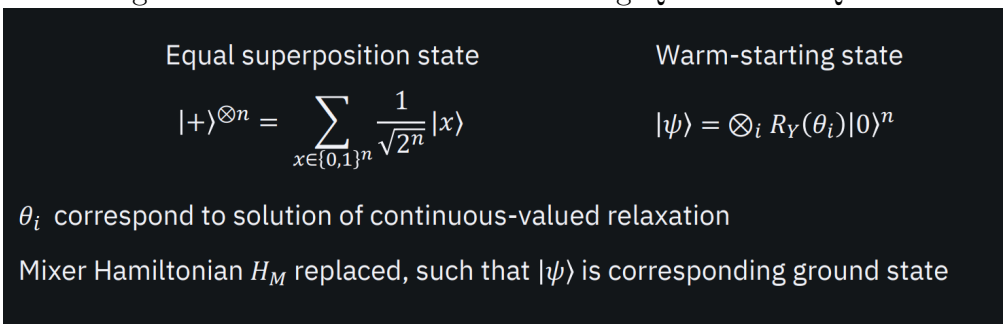
3.2.2 Warm-starting QAOA

Egger, Marecek and Woerner proposed warm-starting QAOA in 2021.[5] Warm-starting QAOA can be simply described as starting with some good but not easy states. It replaces initial state of QAOA with a state obtained by classical optimization algorithms. Recall in the original QAOA, we should start with a simple eigenstate corresponding to the highest/lowest energy state. In this project, we begin with state $\bigotimes_{i=1}^{n} |+\rangle$ which is the highest energy state of Hamiltonian $H_M = \sum_{i=1}^{n} \sigma_i^x$. In warm-start QAOA, we start with more complex energy state. We first relax the binary constraint of our defined problem. The parameters of the initial state are determined by solving the relaxed problem in classical way. After that, we prepared our initial state and mixer layer based on the parameters we get. The cost Hamiltonian is still the same.

$$|+\rangle^{\otimes n} = \sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^n}}|x\rangle \Rightarrow |\psi\rangle = \otimes_i R_Y(\theta_i)\,|0\rangle^n \tag{17}$$

Figure 12,13 shows the experimental result. The two sets of experiment are running with

Figure 11: Notes about Warm-starting QAOA from Qiskit



the same condition except one uses the original QAOA while the other one uses WS-QAOA. The result shows we get higher probability for our best solution in WS-QAOA, which means WS-QAOA can improve the optimization process and accelerate the speed of convergence.

Figure 12: Probability Distribution of QAOA

```
['101100: value: 23.000, probability: 0.2%',
 '010011: value: 23.000, probability: 0.2%',
 '110100: value: 22.000, probability: 1.2%',
 '011100: value: 22.000, probability: 3.6%',
 '100011: value: 22.000, probability: 3.6%',
 '001011: value: 22.000, probability: 1.2%',
 '110010: value: 21.000, probability: 0.5%',
 '011010: value: 21.000, probability: 5.8%',
 '100101: value: 21.000, probability: 5.8%',
 '001101: value: 21.000, probability: 0.5%']
```

Figure 13: Probability Distribution of WS-QAOA

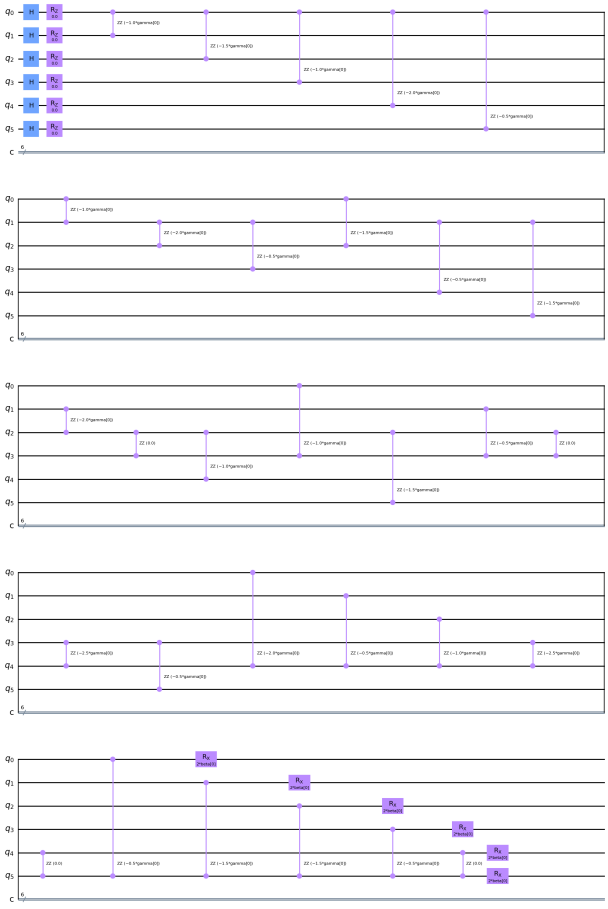
```
['101100: value: 23.000, probability: 0.0%',
 '010011: value: 23.000, probability: 11.1%',
 '110100: value: 22.000, probability: 0.3%',
 '011100: value: 22.000, probability: 0.3%',
 '100011: value: 22.000, probability: 2.0%',
 '001011: value: 22.000, probability: 1.4%',
 '110010: value: 21.000, probability: 4.0%',
 '011010: value: 21.000, probability: 2.5%',
 '100101: value: 21.000, probability: 0.4%',
 '001101: value: 21.000, probability: 0.2%']
```

4 Challenge and Acknowledgment

One challenge of this project is that it's easier to mix up the sequence of the qubits. Qiskit outputs the sampling bitstring in a reversed order by default. At first, I did not realize this problem, and it took me hours to find and fix this problem.

At the very beginning of this project, I had no idea of where to start my QAOA research. I spent many days in reading the original paper of QAOA, but that paper was obviously not appropriate for learning implementation of QAOA. Thanks to Qiskit Learning Community, I quickly knew how to implement it after taking some courses from it. It also provides many well-written functions for doing this QAOA research which definitely increase my efficiency. In fact, some code used in this project was modified from Qiskit's QAOA teaching materials. I also got familiar with Qiskit programming with the help of these learning materials. This is another time I benefit from the spirit of open science. I am also grateful to those who devote to spreading the knowledge of quantum computing. Their suggestions and ideas help me finish this project.

Figure 14: The QAOA Circuit Simulated in This Project



References

[1] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[2] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv preprint arXiv:1411.4028*, 2014.

[3] R. Shaydulin, K. Marwaha, J. Wurtz, and P. C. Lotshaw, "Qaoakit: A toolkit for reproducible study, application, and verification of the qaoa," in *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*, pp. 64–71, 2021.

[4] P. K. Barkoutsos, G. Nannicini, A. Robert, I. Tavernelli, and S. Woerner, "Improving variational quantum optimization using cvar," *Quantum*, vol. 4, p. 256, 2020.

[5] D. J. Egger, J. Mareček, and S. Woerner, "Warm-starting quantum optimization," *Quantum*, vol. 5, p. 479, 2021.