Sean Mcilvane

QAOA for an Optimal Portfolio of Quantum Computing Stocks

1. Introduction

In this paper, I use the hybrid Quantum Approximate Optimization Algorithm (QAOA) to find an optimal stock portfolio of public quantum computing companies. The QAOA was created by (1) and is used to find approximate solutions to difficult combinatorial problems. I will be using the hybrid QAOA to find the combination that maximizes expected returns at a given expected risk.

In a hybrid approach, both classical and quantum computers are used to enhance the performance of the algorithm. The quantum computer and an initial set of parameters are used to create a trail state. The initial state of the QAOA is an equal superposition on all qubits, $|+\rangle^{\otimes N}$. Then, two different Hamiltonians alternate being applied onto the initial state. A cost Hamiltonian H_C encodes the cost function into the trail state and a mixing Hamiltonian H_M is used to explore the subspace. The Hamiltonians alternate being applied for p layers. The duration of applying H_C and H_M is controlled by the parameters γ and β, respectively. The trail state is created by (1):

$$|\Psi_t\rangle = e^{-i\beta_p H_M} e^{-i\gamma_p H_C} \dots e^{-i\beta_2 H_M} e^{-i\gamma_2 H_C} e^{-i\beta_1 H_M} e^{-i\gamma_1 H_C} |+\rangle^{\otimes N}$$

The parameterized quantum trial state prepared and then measured in the Z basis repeatedly to calculate the expectation value of the cost function on the trail state where:

$$\langle H_P \rangle = \langle \Psi_t | H_C | \Psi_t \rangle$$

The most frequently measured bitstring is the approximate solution proposed by the QAOA. A classical optimizer is used to generate new parameters γ and β to minimize the expectation value of the cost function $\langle H_C \rangle$ in hopes of finding a better approximate solution and the whole process is repeated until a satisfying solution is found.

2. Experiment Setup

For this experiment, a portfolio of stocks was selected from a basket of 5 quantum computing stocks which are $ARQQ, $AMZN, $GOOGLE, $IBM, and $IONQ. The expected return of a stock was calculated by the average daily change in the stock's price from October 1, 2021 to Dec 3, 2021. The risk of stock is based on its correlation to other picked stocks. A positive correlation between two stocks means that when one stock drops in price, the other stock drops in price too. A negative correlation between two stocks means that when one stock drops in price, the other stock typically rises in price, and vice versa. A great way to reduce risk in a portfolio is to ensure that some of the stocks in a portfolio have a negative correlation or no correlation. This significantly decreases the probability that all your stocks in your portfolio will drop in value at the same time which would cause a large and terrifying drop in your portfolio's value. Having some stocks with a negative correlation or no correlation reduces the volatility of your portfolio and therefore reduces risk. σ is the covariance matrix where an element describes the correlation between two stocks (Ex: σ_{12} is the correlation between stock 1 and stock 2, which in this paper would be the correlation between $AMZN and $ARQQ).

I created a QAOA using Qiskit that is designed specifically for this experiment where the goal is to minimize the cost function which I modeled as $C = -(1-q)\vec{\mu}\vec{x}^{\,T} + q\vec{x}\sigma\vec{x}^{T}$ based on the Markowitz model (3). The risk factor is q where $q \in \mathbb{R} \mid 0 \leq q \leq 1$ and it controls whether low risk is favored ($q = 1$) or if higher returns is desired ($q = 0$). σ is the covariance matrix and $\vec{\mu}$ is the mean return vector. The choice of stocks in the portfolio is represented by $\vec{x} = [x_1, \ldots, x_N]$ with $x_i \in \{0, 1\}$, where $x_i = 0$ means the stock is not chosen to be in the portfolio and $x_i = 1$ means the stock is chosen to be in the portfolio. The mixing Hamiltonian is designed to explore all possible combinatorial solutions based on (1). The cost Hamiltonian was specifically designed for this experiment and is based on the cost function. The mixing Hamiltonian is:

$$H_M = (X_0 \otimes I_1 \ldots \otimes I_N) + (I_0 \otimes X_1 \ldots \otimes I_N) + \cdots + (I_0 \otimes I_1 \ldots \otimes X_N)$$

Since individual terms commute, the mixing Hamiltonian is applied as:

$$U(H_M) = e^{-i\beta_p H_M} = e^{-i\beta_p X_0} e^{-i\beta_p X_1} \ldots e^{-i\beta_p X_N}$$

The cost Hamiltonian is:

$$H_C = q \sum_{i=0}^{N} \sum_{j \neq i}^{N} \sigma_{ij} Z_i Z_j - (1-q) \sum_{i}^{N} \mu_i Z_i$$

And is applied as:

$$U(H_C) = e^{-i\gamma_p H_C} = \prod_{i=0}^{N-1} \prod_{j \neq i}^{N-1} e^{-i\gamma_p q \sigma_{ij} Z_i Z_j} \ldots \prod_{i=0}^{N-1} e^{i\gamma_p (1-q) \mu_i Z_i}$$

Putting it all together with state preparation, the cost Hamiltonian, and the mixing Hamiltonian, an example of the ansatz looks like this:

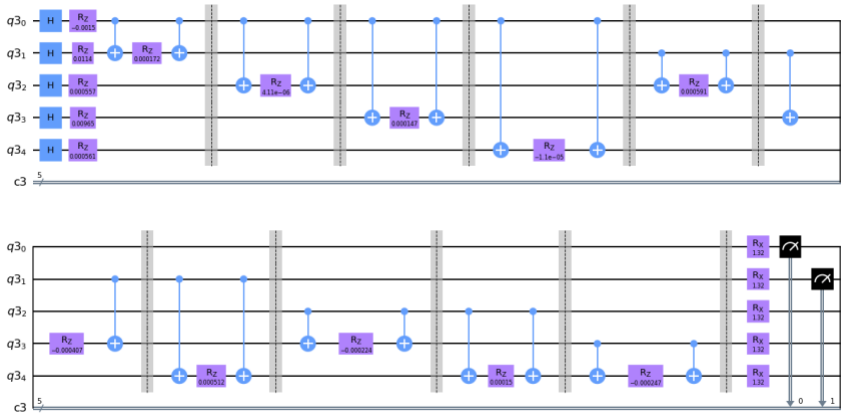
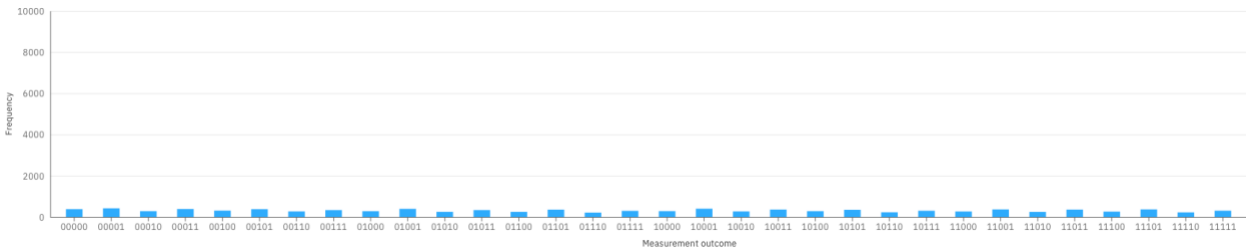


Figure 1 above is an organized version of the ansatz that makes the separate parts clearer. The actual version of the ansatz used in the experiment utilizes many SWAP gates to do the same thing as the above figure with only nearest-neighbor qubit connectivity (see Appendix A). The qubits are first put into equal superposition. The cost Hamiltonian is applied and then the mixing Hamiltonian is applied. All the qubits are then measured in the Z basis. The measurements are used to calculate the expectation value of the cost Hamiltonian on the trail

state. The most frequently measured 'bitstring' is the approximate solution that the QAOA proposes. The parameters used in the cost Hamiltonian to generate a trail state are the updated by a classical optimizer such that the next generated trail state will lead to a lower expectation value of the cost Hamiltonian. This experiment uses the COBYLA (Constrained optimization by linear approximation method).

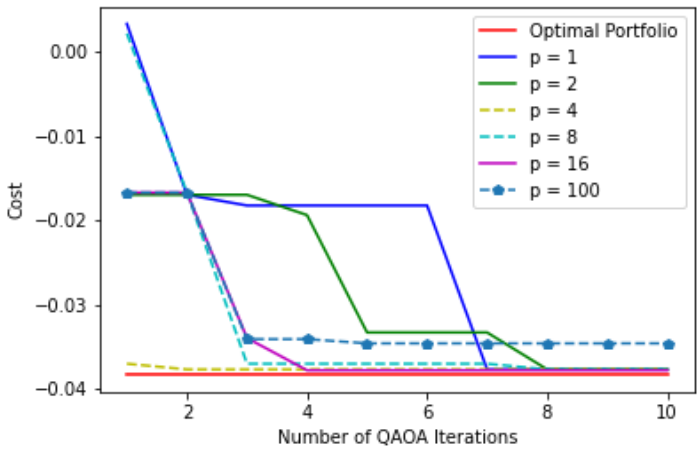
Results.

The main goal of the QAOA is not to find the best solution, but rather find a 'good enough' solution quickly. In this experiment, the solution (portfolio of stocks) that leads to the lowest cost function is the best. I use the classical NumPyMinimumEigenSolver to find the best portfolio so that I can benchmark the QAOA results against it. The objective of the QAOA is to minimize the expectation value of the cost Hamiltonian on the trial state and the solution that is chosen is the bitstring that is measured the most. Below (Figure 2) is a histogram of measurements done a real quantum computer. I used IBM's quantum computer, Bogota, which features 5 qubits that are connected by nearest-neighbors which made it the best fit to run the circuit on.



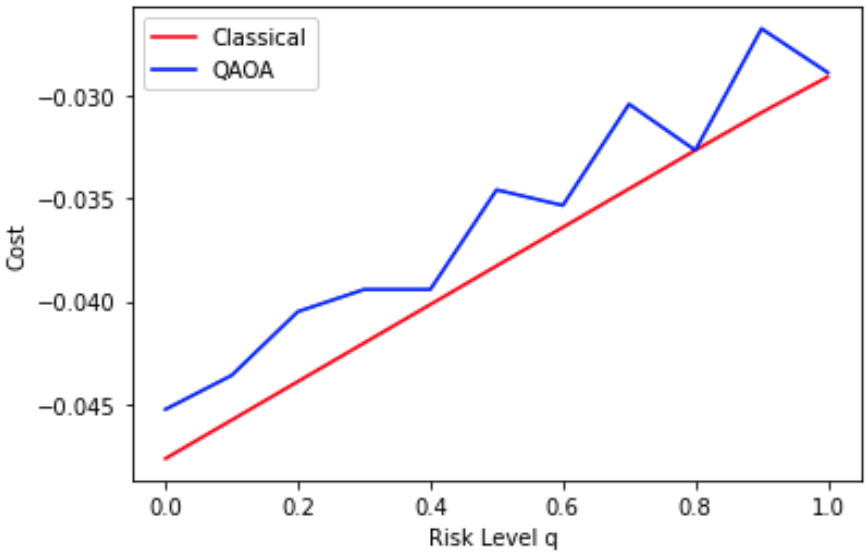
As you can see, the states differ by only a relatively few measurements, which is why it is important to use many shots (repetitions of preparing and measuring a trial state) so that the best bitstring can be identified easier. The bitstring '10001' is measured the most frequently and therefore this would be the solution proposed by the QAOA.

Theoretically, as the number of layers $p \rightarrow \infty$, the accuracy of the algorithm increases (1). Below is Figure 3 showing how the number of layers affected the quality of the approximation made by the QAOA.



As the number of layers used, p, increases, the performance of the QAOA is better with only $p = 4$ layers not following the trend. While the QAOA will eventually converge to a good approximate solution, using more layers can allow for faster convergence. One limit to the number of layers that can be used is quantum volume, as the number of layers increases, the circuit depth increases as well, which allows errors to add up and higher chance of state decoherence.

Below is Figure 4 showing the cost of the best solution that 10 iterations of the QAOA found with $p = 4$ layers at different values of risk, q. Generally, the approximate solutions found by the QAOA are relatively close to the optimal solutions calculated classically. Considering that there are $2^5 = 32$ different possible combinations of stocks in the portfolio, finding close, approximate solutions with only at most 10 iterations of the QAOA is great.



The best portfolio looked different at different risk values. At $q = .5$, the optimal bitstring was '11101', which corresponds to a portfolio of $AMZN, $ARQQ, $GOOGL, and $IONQ. However, with the least risk at $q = 1$, the optimal bitstring is '11001' which corresponds to a portfolio of $AMZN, $ARQQ, and $IONQ.

Discussion.

The QAOA can find good approximations to complex, combinatorial problems when finding the best solution is too expensive. Work has been done on using the Quantum Alternating Operator Ansatz to restrict the subspace with constraints to speed up finding a good solution (2). The Quantum Alternating Operator Ansatz places constraints on the possible subspace by modifying the mixing Hamiltonian (2). The Quantum Alternating Operator Ansatz has been used in portfolio optimization to constrain the number of assets that could be picked for the portfolio (3). While restricting the subspace is great for faster and more accurate approximate solutions, the goal of my experiment was to find a good, approximate solution for an optimal portfolio across the whole subspace of 32 different combinations of 5 stocks. The QAOA in this paper provided a good, approximate solution by the third iteration most of the time as demonstrated in Figure 3.

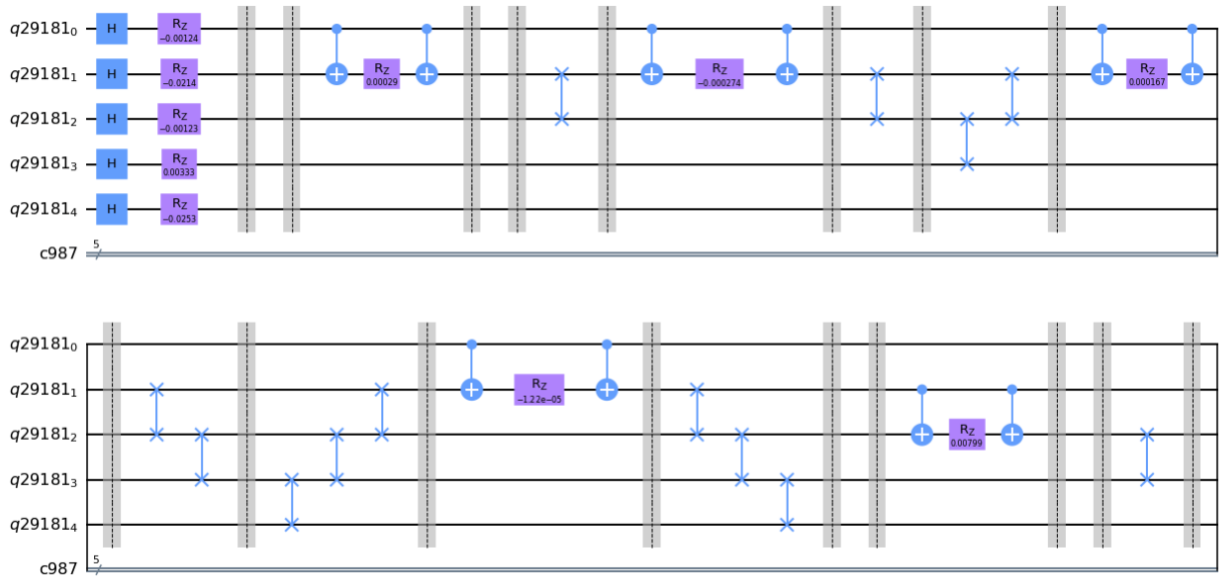
References.

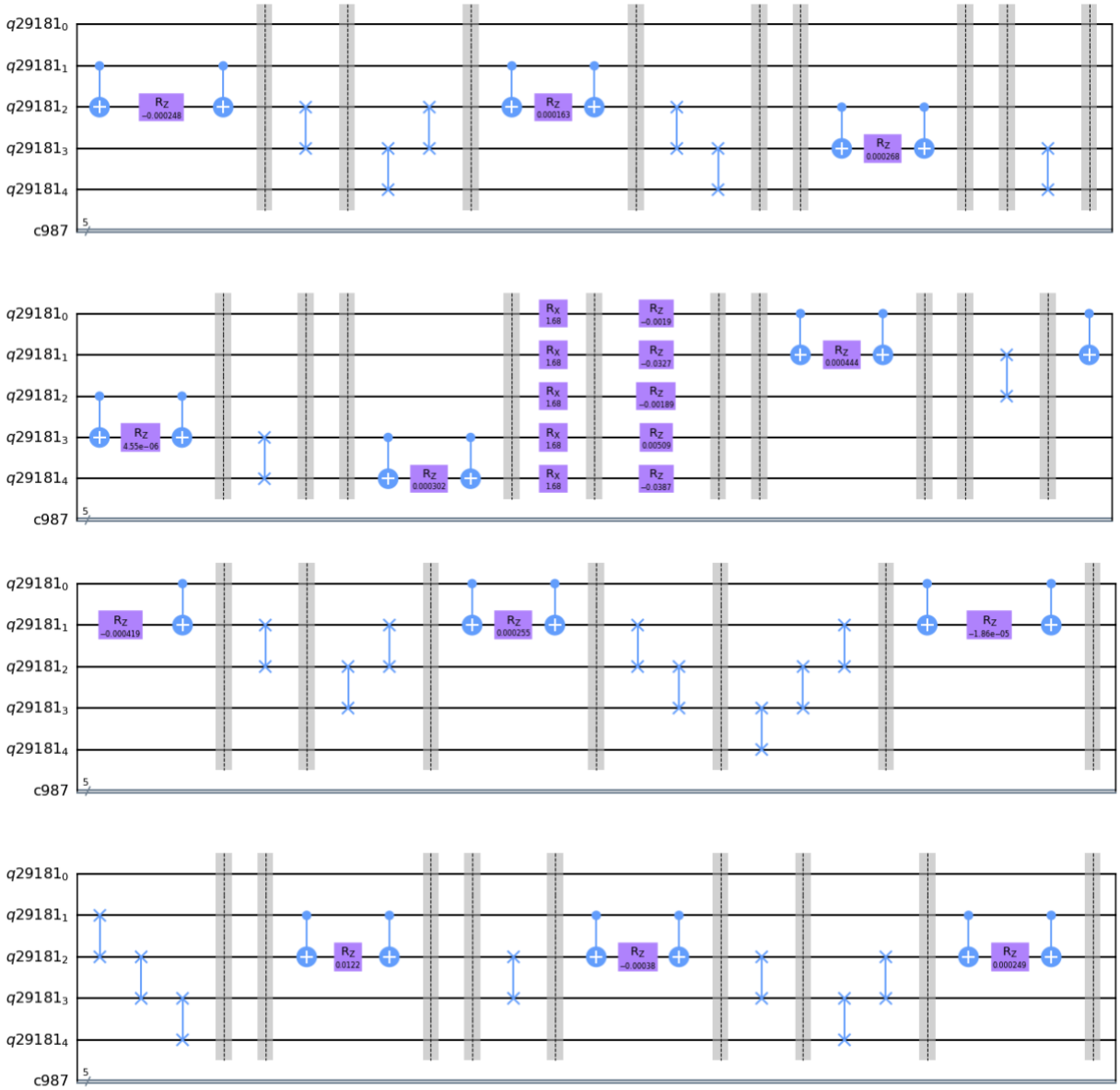
[1] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," 2014. arXiv: 1411. 4028 [quant-ph]. [Online]. Available: https://arxiv. org/abs/1411.4028.

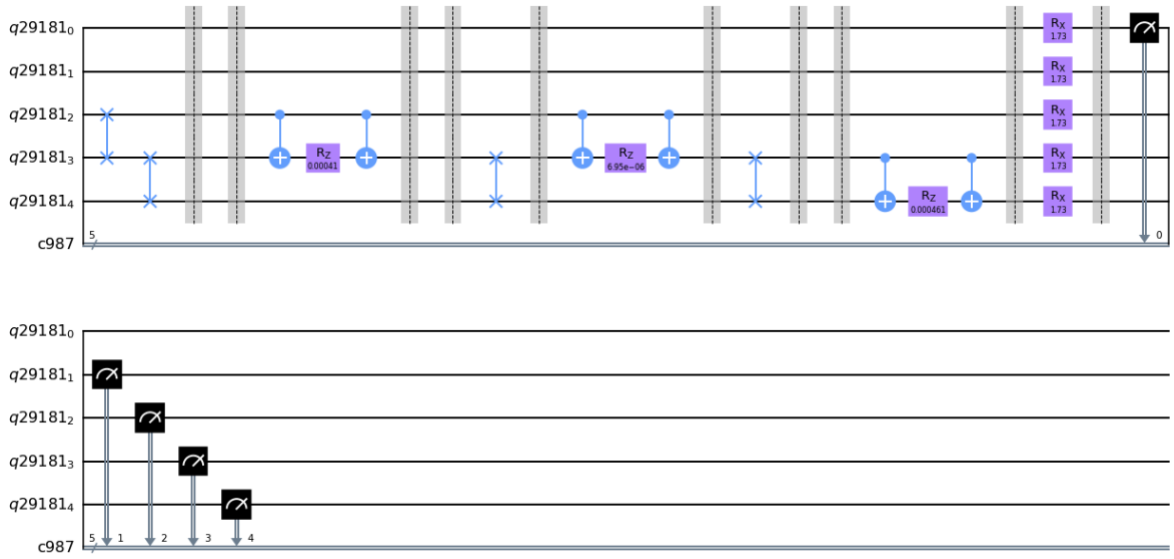
[2] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952, ISSN: 1540- 6261. DOI: 10.1111/j.1540-6261.1952.tb01525.x. [Online]. Available: http://dx.doi.org/10.1111/j.1540- 6261.1952.tb01525.x.

[3] M. Hodson, B. Ruck, H. Ong, D., and Garvin, Stefan D., "Portfolio rebalancing experiments using the Quantum Alternating Operator Ansatz," 2019.arXiv: 1911.05296v1 [quant-ph]. [Online]. Available: https://arxiv.org/pdf/1911.05296.

Appendix A. Nearest-Neighbor Ansatz with 2 layers.




q29181₀
q29181₁
q29181₂
q29181₃
q29181₄
c987
5
Rz −0.000248
Rz 0.000163
Rz 0.000268
Rz 4.55e−06
Rz 0.000302
Rx 1.68
Rx 1.68
Rx 1.68
Rx 1.68
Rx 1.68
Rz −0.0019
Rz −0.0327
Rz −0.00189
Rz 0.00509
Rz −0.0387
Rz 0.000444
Rz −0.000419
Rz 0.000255
Rz −1.86e−05
Rz 0.0122
Rz −0.00038
Rz 0.000249



Appendix B. Link to the Code.

The code for the QAOA can be accessed at
https://github.com/seanmcilvane/QuantumETFBalance