QC803: QAOA, Portfolio Optimization

André Gomes, Martín Marcuello

December 28, 2025

1 Introduction

Since the early twenty-first century, quantum computing has motivated the search for algorithms that outperform classical methods by exploiting uniquely quantum effects, particularly for optimization problems with combinatorial complexity. Hybrid quantum–classical approaches such as the Quantum Approximate Optimization Algorithm (QAOA) aim to address these challenges on near-term hardware by mapping problems to quantum Hamiltonians and iteratively improving solutions. This work investigates portfolio optimization as a case study, formulating it as a quadratic binary optimization problem and assessing the potential benefits and current limitations of quantum optimization methods.

2 Problem Set Up and formulation

2.1 QUBO Formalism

In order to implement our problem on a quantum architecture, it must be reformulated as a QUBO (Quadratic Unconstrained Binary Optimization) problem. A QUBO is a mathematical optimization framework in which the objective is to minimize (or maximize) a quadratic cost function defined over a vector of binary decision variables. The standard formulation of a QUBO objective function is (1):

$$q(x) = \sum_{j=1}^{n} Q_{jj} x_j + \sum_{\substack{j,k=1 \\ j<k}}^{n} Q_{jk} x_j x_k = \mathbf{x}^T Q\, \mathbf{x} \tag{1}$$

where $x = (x_1, x_2, \cdots, x_n) \in \{0, 1\}$ is a vector of binary variables, and Q is a symmetric matrix that defines the coefficients of the objective function. This representation allows encoding both the individual costs associated with each variable and the interactions between pairs of variables.

To adapt the portfolio optimization problem to a QUBO formulation, we encode it as follows. Each asset is assigned to a component of the binary variable vector (qubit). A value of "1" indicates that the corresponding asset is selected for investment, while a value of "0" indicates that it is not selected. Moreover, for the coefficents of the cost function, an expected return vector $\mu \in \mathbb{R}^n$ and a covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$ will be used. In order to control the trade-off between maximizing expected return and minimizing the risk, the risk aversion parameter q will be used.

Once all the parameters are correctly implemented, a penalty constraint will be included in order to limit the number of assets that are chosen. This penalty has de form (2).

$$P = \lambda \left(\sum_{i=1}^{n} x_i - B \right)^2 \tag{2}$$

where B is the total budget (number of assets). The final expresion of the cost function is (3)

$$q(\mathbf{x}) = -\sum_{j=1}^{n} \mu_j x_j + q \cdot \sum_{\substack{j,k=1 \\ j<k}}^{n} \Sigma_{ij} x_j x_k + \lambda \left(\sum_{i=1}^{n} x_i - B \right)^2 \tag{3}$$

2.2 QAOA from the QUBO representation

Once the QUBO problem is stated, a reformulation has to be done in order to fit in a gate-based formalism. For this purpose, the Quantum Approximate Optimization Algorithm (QAOA) is adopted. QAOA is motivated by the adiabatic theorem, which states that a quantum system governed by a time-dependent Hamiltonian will remain in its instantaneous ground state provided that the evolution is sufficiently slow and the system is initially prepared in the ground state. The Hamiltonian will be:

$$\hat{\mathcal{H}}(t) = \left(1 - \frac{t}{\tau}\right)\hat{\mathcal{H}}_0 + \frac{t}{\tau}\hat{\mathcal{H}}_1, \quad (0 \leq \tau \leq t) \tag{4}$$

$$\hat{\mathcal{H}}_0 = -\sum_i \hat{X}_i \tag{5}$$

$$\mathcal{H}_1 = -\sum_{i=1}^{n} h_i Z_i - \sum_{\substack{i,j=1 \\ i<j}}^{n} J_{ij} Z_i Z_j, \tag{6}$$

where $\mathcal{H}_1$ is the ising hamiltonian, analogous to the QUBO cost function. Then, the hamiltonian will be trotterized. The goal is to tune the parameters $\vec{\gamma} = (\gamma_1, \gamma_2, ..., \gamma_p)$ and $\vec{\beta} = (\beta_1, \beta_2, ..., \beta_p)$ for finding the state $\left|\vec{\gamma}, \vec{\beta}\right\rangle = \prod_{k=1}^{p} e^{-i\beta_k \hat{H}_m} e^{-i\gamma_k \hat{H}_c} |+\rangle^{\otimes n}$ that encodes the bitstring that minimices the cost function.

2.3 Metrics

Before proceeding, it is crucial to define two key metrics that were extensively used to analyze the simulated data throughout this project.The first is the approximation ratio, denoted as $C(z)$, which is defined as:

$$C(z) = \frac{C_{max} - C_{obj}}{C_{max} - C_{min}}$$

where C_{max} represents the global maximum (the worst possible cost), C_{min} is the global minimum (the optimal cost), and C_{obj} is the objective value found by the algorithm in question (in this case, either QAOA or the Greedy solver). A ratio of 1 indicates the algorithm found the global minimum, while 0 corresponds to the worst-case scenario, values closer to 1 indicate higher quality solutions.

The second critical metric is the Success Probability. It offers a stricter evaluation by measuring the likelihood of observing the exact optimal bitstring in the QAOA output. Unlike $C(z)$, which can be high even for sub-optimal solutions with good energy, this metric specifically assesses the algorithm's precision in pinpointing the true global minimum rather than a close approximation.

3 Results and Discussion

Before proceeding further, it is important to note that, although significant efforts were made to ensure deterministic behavior across simulations (e.g., by fixing random seeds), slight variations persist upon re-execution. This inherent stochasticity implies that the conclusions drawn herein

should be interpreted with a degree of caution. The observed variability is driven by numerous parameters and internal optimization dynamics that, while impactful, fall outside the scope of this project.

3.1 Data Collection and Preparation

We now turn our attention to the experiments. First, the expected return vector μ and the covariance matrix Σ for the year 2016 were retrieved from Yahoo Finance using the `get_portfolio_data` function. Subsequently, all entries of the vector μ were normalized to the range $[0, 1]$, and the same normalization was applied to the diagonal of the matrix Σ. Finally, using the `create_portfolio_qp` function, an object of the QuadraticProgram class was instantiated. This class is utilized by Qiskit to apply the QAOA algorithm to a specific portfolio instance.

3.2 QAOA vs Greedy Analysis

The setup for this analysis is: $n = 16$, $q = 0.9$, $p = 3$, $B = 8$, $\lambda = 16$.

The performance of QAOA using three distinct optimizers (COBYLA, SPSA, POWELL) was benchmarked against a classical greedy algorithm, as summarized in Table 1

Table 1: Performance comparison of QAOA optimizers and the classical Greedy solver.

Optimizer	**Ratio** $C(z)$	**Value**	**Best String**	**Success Prob**
COBYLA	0.9931	-0.2555	0010001101111010	0.0000
SPSA	0.9931	-0.2555	0010001101111010	0.0000
POWELL	0.9931	-0.2555	0010001101111010	0.0000
Greedy	0.7520	0.4617	1100101001101010	N/A

As observed, the Classical Greedy Solver underperformed (Ratio 0.7520) by ignoring covariance in Σ. In contrast, all three QAOA optimizers consistently converged to a superior sub-optimal solution (Ratio 0.9931). Although QAOA failed to identify the global optimum (Success Probability 0), it significantly outperformed the greedy approach in overall solution quality.

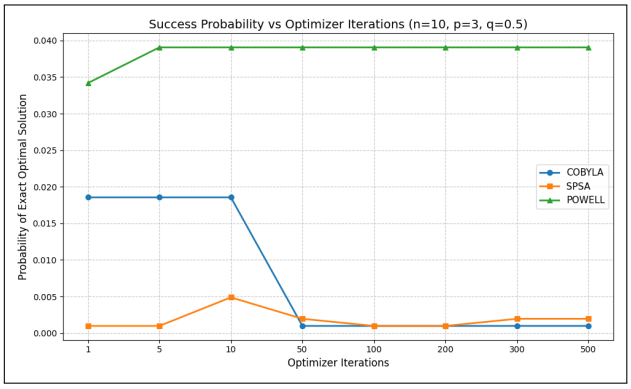
3.3 Optimizer Analysis

The setup for this analysis is: $n = 10$, $q = 0.5$, $p = 3$, $B = 5$, $\lambda = 10$.

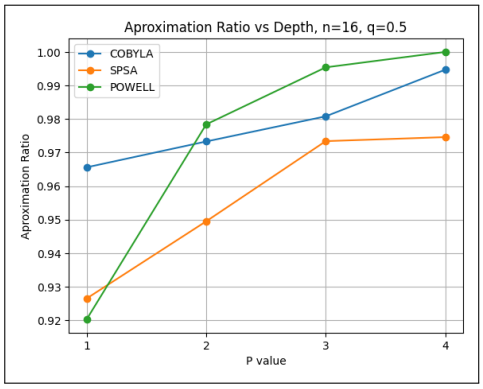
One of the parameters we can control in the optimizers is the number of iterations allowed to converge to a minimum. The results of this analysis for the three different optimizers are shown in Figure 1a. It is possible to observe in Figure 1a that the POWELL optimizer is generally more accurate than SPSA or COBYLA, as the probability of finding the exact optimal solution is significantly higher compared to the other optimizers.

3.4 Depth Analysis (p)

The setup for this analysis is: $n = 16$, $q = 0.5$, $B = 8$, $\lambda = 8$. In order to check how the QAOA algorithm performs we can tune the depth of the circuit, ie. tuning the number of alternating layers of cost and mixer unitaries applied in the QAOA circuit. The results are shown in Figure 1b.



(a)



(b)

Figure 1: (a) Optimizer Analysis Results, (b) Depth Analysis results

It is noticeable that the higher the Depth of the circuit is, the better the performance of the QAOA. However, increasing the p value can cause some problems. Increasing p improves the solution quality in principle, but it also increases the circuit depth, and therefore the noise sensitivity. These are the main reasons why small p values more practical on near term (NSIQ) devices.

3.5 Penalty Analysis (λ)

The setup for this analysis is: $n = 10$, $q = 0.5$, $p = 3$, $B = 5$.

By varying the penalty factor λ while holding all other parameters constant, it is possible to analyze its impact on the Success Probability.

One can conclude from Figure 2a that there is an optimal 'sweet spot' for the penalty value. It can be neither too high nor too low to achieve maximum success probability.

3.6 Noise Analysis

The setup for this analysis is: $n = 4$, $q = 0.5$, $p = 2$, $B = 2$, $\lambda = 4$.

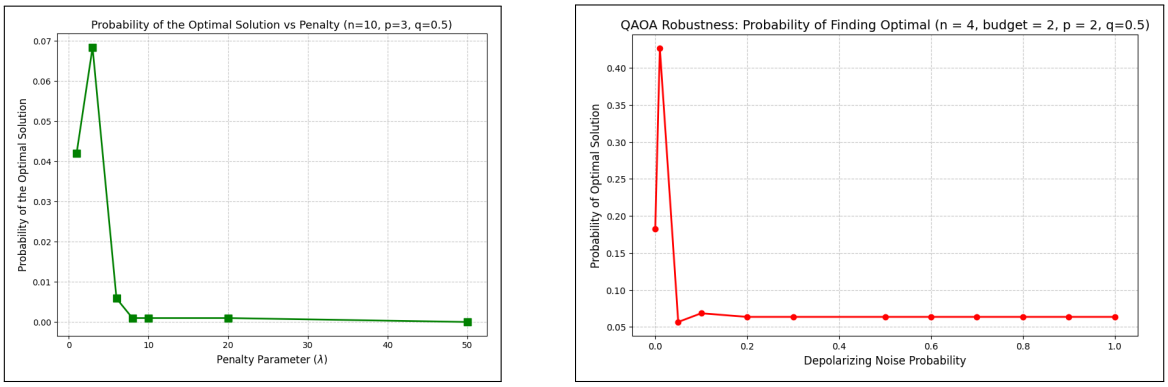
To conduct the noise analysis, an error probability was assigned to each gate in the circuit created by Qiskit. The results are presented in Figure 2b.

As shown in Figure 2b, a small amount of depolarizing noise can lead to marginally improved performance compared to the noiseless case. This effect is attributed to stochastic and optimization-related factors rather than a fundamental advantage of noise.

Nevertheless, the overall trend shows that increasing the depolarizing noise probability degrades QAOA performance. As the noise level increases, decoherence increasingly suppresses the quantum correlations required for the algorithm, leading to a systematic reduction in solution quality.

4 Conlcusions

In this work, we investigated the application of the *Quantum Approximate Optimization Algorithm* (QAOA) to the portfolio optimization problem, formulated within the QUBO framework. Through



(a) (b)

Figure 2: (a) Penalty Analysis Results, (b) Noise Analysis Results

classical simulations, we analyzed the performance of QAOA under different parameter regimes and compared it against a classical greedy approach.

The results demonstrate that QAOA is capable of producing high-quality solutions, consistently outperforming the greedy solver in terms of approximation ratio. This highlights the ability of variational quantum algorithms to capture complex correlations.

Our analysis shows that the choice of classical optimizer plays a crucial role in the overall performance of QAOA. Among the optimizers considered, POWELL exhibited superior accuracy and a significantly higher success probability compared to SPSA and COBYLA.

Additionally, the depth analysis confirmed that increasing the number of QAOA layers p generally improves solution quality, albeit at the cost of increased circuit depth and greater sensitivity to noise. This trade-off limits the practicality of large p values on near-term NISQ devices.

The study of the penalty parameter λ revealed the existence of an optimal "sweet spot." If λ is too small, the budget constraint is insufficiently enforced, while excessively large values distort the energy landscape and hinder convergence.

Furthermore, the noise analysis shows that increasing the depolarizing noise consistently degrades QAOA performance, reducing the algorithm's reliability and solution quality.

Overall, this project illustrates both the promise and the current limitations of QAOA for financial optimization problems. QAOA it is still an open area of research and there has not been still a demonstration of fully quantum advantage. Farhi et al. showed that level p = 1 QAOA is computationally hard to simulate on a classical computer without collapsing the polynomial Hierarchy. However, this does not imply that QAOA for p = 1 is able to solve problems more efficiently than classical computers. Another interesting remark is the choosing of the initial set of angles. Brandao et al. (Brandao et al., 2018) demonstrated that if the problem instances come from a reasonable distribution, then the expectation value of the cost function concentrates. This suggests that it is possible to train a classical optimizer to find good angles on small instances and reuse those angles on larger instances, as long as they come from the same distribution.

References

[1] Fernando G. S. L. Brandão et al. *For Fixed Control Parameters the Quantum Approximate Optimization Algorithm's Objective Function Value Concentrates for Typical Instances*. https://arxiv.org/abs/1812.04170. arXiv:1812.04170. 2018.

[2] Edward Farhi et al. *Quantum Algorithms for Fixed Qubit Architectures*. https://arxiv.org/abs/1703.06199. arXiv:1703.06199. 2017.

[3] M. Hernández and otros. *Quantum Computing for Finance: Portfolio Optimization*. https://arxiv.org/abs/2207.10555. arXiv:2207.10555. 2022.

[4] Erisk Kokkoni. *Lecture notes on quantum computing*. Lecture Notes, Chalmers University of Technology. Quantum Computing. 2025.

[5] Qiskit Community. *Portfolio Optimization with Qiskit Finance*. https://qiskit-community.github.io/qiskit-finance/tutorials/01_portfolio_optimization.html. [Online tutorial]. n.d.