Hardware Specific Error-Correction and Noise Resistant Portfolio Optimization with Variational Quantum Eigensolver

Castellanos-Cubides, M.[1], Chen-Adamczyk, I.[1], Hussain, H.[1], Li, D.[1], Girmay, T.[1], and Zhen, Y.[1]

[1]Powered by *QBraid*

September 28, 2024

Abstract

In this project, we integrate a custom Variational Quantum Eigensolver (VQE) into portfolio optimization to enhance scalability and performance on noisy intermediate-scale quantum (NISQ) devices. We introduce a Shot-Efficient Simultaneous Perturbation Stochastic Approximation (SE-SPSA) algorithm, a modification of the classical SPSA, which dynamically adjusts the number of measurement shots during optimization. This approach effectively balances resource efficiency and accuracy, making it suitable for the constraints of current quantum hardware. By mapping the portfolio optimization problem onto a quantum Hamiltonian, we leverage quantum algorithms to find optimal asset allocations that balance risk and return. We implement hardware-specific optimizations, including dynamic decoupling and qubit mapping, to mitigate noise and decoherence effects. Our results demonstrate that the combination of custom VQE and SE-SPSA effectively navigates the noisy landscape of NISQ devices, offering promising avenues for quantum-enhanced financial algorithms.

Keywords: Quantum Portfolio Optimization, Variational Quantum Eigensolver, SE-SPSA, Quantum Computing, Financial Algorithms, Error Correction, Noise Reduction.

Columbia University
in the City of New York

NYU

1 Introduction

Portfolio optimization is a critical task in finance, aimed at balancing risk and returns across asset allocations. Traditional methods, such as the Markowitz mean-variance framework, struggle with high-dimensional data and complex constraints, leading to significant computational demands that limit their scalability. Quantum computing, with its ability to explore multiple solutions simultaneously through superposition and entanglement, offers a promising alternative for addressing these challenges. However, current noisy intermediate-scale quantum (NISQ) devices present their own obstacles, such as noise, decoherence, and hardware limitations, which necessitate effective error correction and resource optimization strategies.

This project integrates the Variational Quantum Eigensolver (VQE) into portfolio optimization, focusing on enhancing scalability, feasibility, and performance on noisy quantum hardware. By mapping the optimization problem to a quantum Hamiltonian, we leverage quantum algorithms to find solutions that are computationally intensive for classical methods.

Our approach builds upon methodologies presented in recent studies by Goldman Sachs and AWS [2], and Buonaiuto et al. [1], who have highlighted the potential and challenges of quantum algorithms in portfolio optimization.

2 Method

In this section, we delve into the methods and algorithms we employed to integrate the Variational Quantum Eigensolver into our portfolio optimization problem.

2.1 Data Preparation

We obtained closing price data for the stocks from Yahoo Finance and calculated daily returns. The data spans from June 1, 2022, to September 20, 2024.

```python
import pandas as pd
import numpy as np
from pypfopt import expected_returns, risk_models

# Closing price data obtained from yfinance
# 2year_closing.csv
data = pd.read_csv("close.csv", parse_dates=True, index_col="Date
    ")
```

```

returns_df = data.pct_change().dropna()

frequency = len(returns_df)
mu = expected_returns.mean_historical_return(data, frequency=
    frequency)
S = risk_models.sample_cov(data, frequency=frequency)
```

Listing 1: Data Preparation

2.2 Mapping to Quantum Hamiltonian

Next, we mapped the portfolio optimization problem onto a quantum Hamiltonian H, where our goal is to find its ground state. The Hamiltonian represents the total energy of the system and includes terms for expected returns, risk (represented by covariances), and penalty terms for budget constraints.

The Hamiltonian is constructed as follows:

$$\mathbf{H} = \sum_i h_i Z_i + r\left(\sum_{i<j} J_{ij} Z_i Z_j\right) + \delta\left(\sum_i p_i Z_i - 1\right)^2$$

where:

- Z_i are Pauli-Z operators acting on qubits.
- h_i represents the expected return coefficients.
- J_{ij} represents the covariance coefficients between assets.
- δ is the penalty factor for the budget constraint.
- p_i are the normalized prices of the assets.
- r is the risk factor of the asset.

We defined functions to calculate these coefficients based on the financial data. For example, to compute the coefficients for expected returns, we used:

```
def rev_the_hamiltonian():
    I = ['I'] * sum(d)

    # Coefficients for expected returns (h_i Z_i)
    h = (np.array([u]) * np.array([P]) / B)
    h_ = []
    hi = []
    for i in range(len(d)):
        for j in range(d[i]):
            h_.append(h[0][i] * (2**j))
            I[i] = 'Z'
            hi.append(''.join(I))
            I[i] = 'I'
```

Listing 2: Computing Expected Return Coefficients

2.3 Quantum Unconstrained Binary Optimization (QUBO)

To represent our optimization problem in a form suitable for quantum algorithms, we utilized the Quantum Unconstrained Binary Optimization (QUBO) model [3]. In the QUBO formulation, we express the objective function as a quadratic polynomial of binary variables:

$$\text{Minimize } f(x) = x^T Q x,$$

where x is a vector of binary variables and Q is a matrix representing the coefficients of the quadratic terms. This formulation allows us to encode both the objective and the constraints of the optimization problem into a single function.

In our project, the binary variables represent the inclusion or exclusion of asset quantities in the portfolio. By incorporating expected returns, covariances, and budget constraints into the QUBO model, we can represent the portfolio optimization problem as a Hamiltonian suitable for quantum algorithms.

2.4 Variational Quantum Eigensolver (VQE)

The Variational Quantum Eigensolver (VQE) is a hybrid quantum-classical algorithm designed to find the ground state (minimum eigenvalue) of a given Hamiltonian [4]. VQE leverages quantum resources to evaluate the expectation value of the Hamiltonian and classical optimization algorithms to adjust the parameters of a parameterized quantum circuit (ansatz) to minimize this expectation value.

The key steps of the VQE algorithm are:

1. **Ansatz Circuit Preparation**: We designed a parameterized quantum circuit using the `EfficientSU2` ansatz provided by Qiskit, which offers a balance between expressibility and circuit depth.

```python
    from qiskit.circuit.library import EfficientSU2

    def cost_funct(params):
        qc = QuantumCircuit(size)
        ansatz = EfficientSU2(size)
        qc.compose(ansatz, inplace=True)
        # Additional circuit operations if needed
```

Listing 3: Ansatz Circuit

2. **Expectation Value Measurement**: Using the quantum circuit, we prepared a quantum state and measured the expectation value of the Hamiltonian.

```python
    from qiskit.primitives import StatevectorEstimator

    def cost_funct(params):
        # Prepare the ansatz circuit
        qc = QuantumCircuit(size)
        ansatz = EfficientSU2(size)
        qc.compose(ansatz, inplace=True)

        # Define the observables (Hamiltonian terms)
        observables = [
            [SparsePauliOp(hi, h_)],
            [SparsePauliOp(ji, J__)],
            [SparsePauliOp(bi, pi_)]
        ]

        # Estimate the expectation value
        estimator = StatevectorEstimator()
        job = estimator.run([qc, observables, params])
        result = job.result()
        return sum(result[0].data.evs)
```

Listing 4: Expectation Value Calculation

3. **Classical Optimization**: We employed a classical optimizer (`COBYLA` from SciPy) to adjust the circuit parameters to minimize the expectation value obtained from the quantum measurements.

```python
from scipy.optimize import minimize

# Initial parameters
x0 = 2 * np.pi * np.random.random(size * 8)

# Optimization
result = minimize(
    cost_funct,
    x0,
    method='COBYLA',
    options={'tol': 1e-4, 'disp': True}
)

print("Optimal parameters:", result.x)
print("Minimum function value:", result.fun)
```

Listing 5: Parameter Optimization

2.5 Resource Usage Analysis

To evaluate the performance and scalability of our implementation, we conducted resource usage analysis by measuring the memory consumption and execution time during the optimization process. This analysis helps us understand the computational demands of the algorithm and its feasibility on current hardware.

2.5.1 Memory Usage Measurement

We used the `memory_profiler` library in Python to monitor the memory usage during the execution of the optimization routine.

```python
from memory_profiler import memory_usage

def ability_test():
    error = 1e18
    while error > 1e8:
        x0 = 2 * np.pi * np.random.random(size * 8)
        result = minimize(cost_func, x0, method='COBYLA', options
           ={'tol': 1e-4, 'disp': True})
        error = result.fun
    v = test_ansatz(result.x)
    return v

with Session(backend=backend) as session:
    size = 5
    shots = 1000
    # Initialize target_vector...
    sampler = StatevectorSampler()
    mem_usage = memory_usage(ability_test, interval=1)
    print("Average memory usage during optimization (MB):", sum(
       mem_usage) / len(mem_usage))
```

Listing 6: Measuring Memory Usage

2.5.2 Execution Time Measurement

We measured the execution time using the `time` module.

```python
import time

with Session(backend=backend) as session:
    size = 5
    shots = 1000
    # Initialize target_vector...
    sampler = StatevectorSampler()
    start_time = time.time()
    ability_test()
    end_time = time.time()
    print("Time elapsed during optimization (s):", end_time -
        start_time)
```
Listing 7: Measuring Execution Time

3 Implementation and Error Correction

Our implementation follows a ranked approach, gradually enhancing the algorithm's performance through a series of targeted improvements:

- **Rank 0: Modeling and Quantization of Portfolio Optimization**
  We began by modeling the portfolio optimization problem within the VQE framework. This involved processing historical financial data to calculate risk, reward, and adherence to budget constraints, and generating a Hamiltonian whose ground state represents the optimal combination of investments. Investments were quantized, ensuring that VQE outputs correspond to whole-number multiples of assets. The cost function was made highly customizable, allowing for variations in the choice of ansatz and simulation services.

- **Rank 1: Classical Simulation of VQE**
  Utilizing the COBYLA minimizer, we simulated the VQE classically to demonstrate convergence on an optimal solution. This step provided a baseline performance level and served as a foundation for subsequent experiments with quantum hardware, helping us understand the theoretical performance of the algorithm without quantum noise interference.

- **Rank 2: Quantum Execution with Hardware Optimization**
  We transitioned from classical simulations to execution on IBM quantum hardware, employing gate translation techniques to match the circuit's gates with those specific to the quantum processor. Qubit mapping was also implemented to select the most coherent and entanglement-capable qubits, minimizing errors and enhancing the expressivity of the VQE circuit. This involved advanced usage of the `pass_manager()` function to rewrite Pauli strings and maintain accurate estimations during the quantum execution.

- **Rank 3: Dynamic Decoupling for Noise Reduction**
  To mitigate noise between qubit operations, dynamic decoupling techniques were introduced. This approach inserted carefully timed sequences during idle periods in the circuit, stabilizing qubits and reducing the impact of decoherence. This enhancement was particularly effective given the structure of our ansatz,

which left relatively long gaps between operations, creating opportunities to preserve qubit states in noisy environments.

- **Rank 4: Shot-Efficient SPSA for Scaling**
  Addressing the challenge of shot scaling—where increasing circuit size exponentially raises the number of shots needed to maintain accuracy—we developed a Shot-Efficient Simultaneous Perturbation Stochastic Approximation (SE-SPSA) algorithm. This algorithm starts with a lower shot count and increases gradually, aligned with iterative accuracy demands. The SE-SPSA approach was tested on Rigetti's "Ankaa-2," showcasing the robustness of our noise-reduction techniques and the ability to adapt shot allocation dynamically, effectively managing the trade-off between accuracy and computational cost.

- **Rank 5: Future Directions in Ansatz Design**
  Although we did not fully implement this rank due to time constraints, we explored the potential for future improvements by challenging the standard supremacy of the Hardware `EfficientSU2` ansatz. Inspired by existing studies [1], we considered whether more complex entanglement structures could outperform the current standard as quantum hardware advances, especially in ion and atom-based systems like QuEra and IonQ. This rank sets the stage for investigating alternative ansatz designs that could leverage enhanced entanglement capabilities, pushing the boundaries of what is possible with VQE in portfolio optimization.

3.1 Shot-Efficient SPSA

The classical SPSA is defined as follows. The Simultaneous Perturbation Stochastic Approximation (SPSA) is an iterative optimization algorithm used to approximate the gradient of a cost function with respect to its parameters.

SPSA works by approximating the gradient using two measurements of the function: one with a positive perturbation and one with a negative perturbation. The gradient estimate is then used to iteratively update the parameters towards the optimal solution. This approach is particularly useful in noisy environments, such as quantum computing, where function evaluations can be resource-intensive and imprecise.

The gradient approximation in SPSA is given by:

$$\nabla f(\vec{\theta}) = \begin{pmatrix} \frac{\partial f}{\partial \theta_1} \\ \vdots \\ \frac{\partial f}{\partial \theta_n} \end{pmatrix} \approx \frac{1}{2\epsilon} \begin{pmatrix} f(\vec{\theta} + \epsilon \vec{e}_1) - f(\vec{\theta} - \epsilon \vec{e}_1) \\ \vdots \\ f(\vec{\theta} + \epsilon \vec{e}_n) - f(\vec{\theta} - \epsilon \vec{e}_n) \end{pmatrix} \approx \frac{f(\vec{\theta} + \epsilon \vec{\Delta}) - f(\vec{\theta} - \epsilon \vec{\Delta})}{2\epsilon} \vec{\Delta}^{-1}$$

In this formula:

- $\vec{\theta}$ represents the vector of parameters.
- ϵ is a small perturbation value.
- $\vec{\Delta}$ is a randomly generated perturbation vector, where each component is independently sampled.

SPSA's simultaneous perturbation of all dimensions allows for efficient gradient estimation with minimal computational overhead, making it well-suited for optimization tasks in noisy environments, such as quantum computing, where function evaluations are costly and noisy.

SE(Shot-Efficient)-SPSA introduces an adaptive mechanism to dynamically adjust the number of shots (measurements) used during optimization. This approach is specifically designed to balance resource efficiency and accuracy, making SE-SPSA highly effective for noisy quantum computers.

How Dynamic Adjustment Works:

- **Initial Stage: Low Shot Count**
  SE-SPSA begins with a lower number of shots per gradient estimation. This early stage focuses on broad exploration of the parameter space, making coarse adjustments with minimal resource consumption. The algorithm evaluates the cost function using fewer shots, which provides a rough approximation of the gradient direction.
- **Adaptive Increase in Shots: Refinement Phase**
  As the optimization progresses, SE-SPSA gradually increases the number of shots based on the iteration number and the convergence behavior. For example, the shot count may increase linearly as $S_k = S_0 + k \times \alpha$, where S_0 is the initial shot count, k is the iteration number, and α controls the scaling rate. This allows for finer parameter adjustments and more accurate gradient estimation as the algorithm nears convergence.
- **Feedback Loop with Convergence Monitoring**
  SE-SPSA continuously monitors the optimization process and adjusts shot counts based on convergence. If the improvement in the cost function diminishes, indicating proximity to the optimal solution, the shot increase slows down to avoid unnecessary resource use. Conversely, if significant parameter adjustments are needed, shot counts are increased to enhance accuracy, allowing the algorithm to better handle noise.

Noise Resistence

- **Resource Efficiency**: By starting with fewer shots and only increasing them as needed, SE-SPSA conserves computational resources, reducing the time and cost associated with quantum measurements.
- **Noise Adaptation**: The stochastic nature of stochastic processes helps the algorithm navigate the noisy landscape of quantum hardware. As shots increase, SE-SPSA reduces the impact of noise, refining the solution with higher precision when necessary.

- **Balancing Exploration and Exploitation**: Early low-shot stages allow broad exploration, while later high-shot stages fine-tune the parameters, ensuring a smooth convergence to the optimal solution. This balance is crucial in noisy environments, where excessive precision early on would waste resources.
- **Scalability**: SE-SPSA's adaptive shot scaling efficiently manages the exponential increase in shot requirements as quantum circuits grow in size, maintaining accuracy without overwhelming the system's capacity.

The dynamic shot adjustment of SE-SPSA makes it particularly effective for quantum optimization, leveraging shot efficiency to overcome the inherent noise and measurement constraints of current quantum computers. By tuning the shot allocation dynamically, SE-SPSA achieves a robust balance between resource use and solution accuracy.

3.2 Results and Performance

This section we will showcase our result.

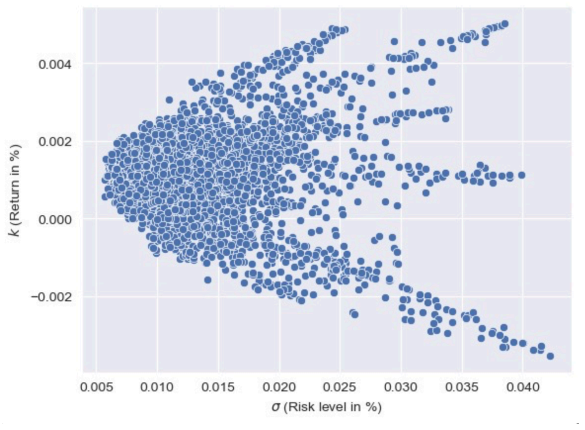

Figure 1: Decision characteristics

Figure 1 illustrates the relationship between risk and return, σ and κ for the portfolios generated by our algorithm. Each point represents a distinct portfolio configuration, showcasing the trade-offs between expected returns and associated risk levels. The data shows a broad spread, indicating diverse portfolio combinations with varying risk-return characteristics.

The dense clustering towards lower risk and return values suggests that most portfolios are conservatively optimized, while the spread towards higher returns with increasing risk reflects more aggressive investment strategies. This visualization helps identify potential optimal portfolios that align with different risk tolerance levels, demonstrating the flexibility and scalability of our VQE-based optimization approach in balancing risk and reward within a noisy quantum environment.

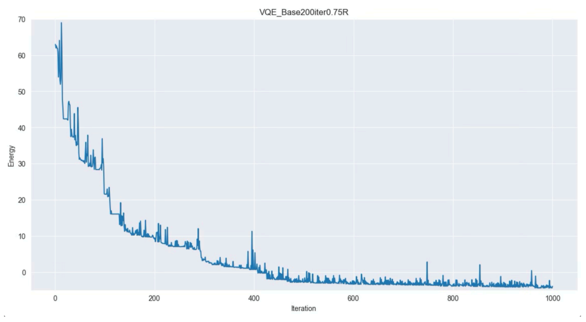

Figure 2: Energy vs Iteration

Figure 2 plots the energy (cost function value) of the VQE algorithm over iterations. The rapid decline in energy demonstrates the algorithm's convergence towards an optimal solution, with significant improvements in the early iterations and finer adjustments later. The gradual reduction and stabilization of the energy value indicate the effectiveness of the SE-SPSA in refining the parameters efficiently.

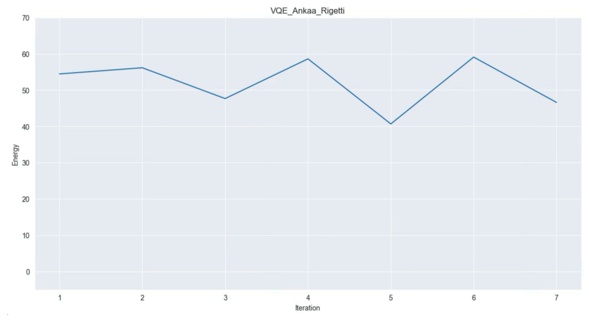

Figure 3: Energy vs Iteration on Ankaa2

Figure 3 shows the energy convergence behavior of the VQE running on Rigetti's Ankaa-2 quantum processor with the SE-SPSA method. The graph illustrates a non-monotonic pattern in the energy values across iterations, indicating fluctuations in convergence. This suggests the presence of noise and variability inherent to current quantum hardware.

The oscillations in energy values highlight the challenges of working with NISQ devices, where SE-SPSA is actively adapting the shot count dynamically to handle noise. Despite these fluctuations, there is an overall downward trend, demonstrating that SE-SPSA effectively manages noise and guides the VQE towards lower energy states.

4 Conclusion

In this project, we successfully integrated the Variational Quantum Eigensolver (VQE) into portfolio optimization, demonstrating its potential to solve complex financial problems that are challenging for classical algorithms. By mapping the portfolio optimization problem onto a quantum Hamiltonian and utilizing a quantum-classical hybrid approach, we were able to find optimal asset allocations that balance risk and return.

Our implementation introduced several innovations to enhance performance on noisy intermediate-scale quantum (NISQ) devices. We developed a Shot-Efficient Simultaneous Perturbation Stochastic Approximation (SE-SPSA) algorithm, which dynamically adjusts the number of measurement shots during optimization to balance resource efficiency and accuracy. This approach effectively managed the trade-off between computational cost and solution precision, making it well-suited for the limitations of current quantum hardware.

Through a ranked series of enhancements, we incrementally improved the algorithm's performance. Starting from classical simulations, we progressed to quantum execution with hardware-specific optimizations, including gate translation and qubit mapping to minimize errors. The implementation of dynamic decoupling techniques further reduced the impact of noise and decoherence, stabilizing qubit states during idle periods in the circuit.

Our results demonstrated that the SE-SPSA method effectively navigates the noisy landscape of NISQ devices, achieving convergence towards optimal solutions in portfolio optimization. The energy convergence plots showed that despite inherent quantum noise, the algorithm could find lower energy states indicative of better portfolio configurations. This indicates that VQE, combined with our SE-SPSA approach, is a promising tool for financial optimization problems.

However, there are limitations to our current approach. The scalability of the algorithm is still constrained by the depth and coherence times of available quantum hardware. As the size of the portfolio (and thus the number of qubits) increases, the circuit becomes more susceptible to noise and errors. Additionally, while our SE-SPSA method improves resource efficiency, further optimization is needed to handle larger, more complex financial models, which was not implemented due to the time constraint of the project, but can be a clear direction for future research.

Future work could focus on exploring alternative ansatz designs that leverage advanced entanglement capabilities of emerging quantum hardware platforms, such as ion-trap or neutral-atom systems. Investigating more sophisticated error mitigation and correction techniques could also enhance the algorithm's robustness. Moreover, integrating real-world financial constraints, such as transaction costs, taxes, and liquidity considerations, would make the model more applicable to practical portfolio management.

In conclusion, our project demonstrates the feasibility and potential advantages of using quantum algorithms, specifically VQE with SE-SPSA, for portfolio optimization. As quantum technology continues to advance, these methods could offer significant improvements over classical approaches, the prospect for quantum portfolio optmization is promisimg.

References

[1] Federico Buonaiuto et al. “Best practices for portfolio optimization by quantum computing, experimented on real quantum devices”. In: Scientific Reports 13.1 (2023), p. 15139. DOI: 10.1038/s41598-023-45392-w. URL: https://www.nature.com/articles/s41598-023-45392-w.

[2] Alexander Dalzell et al. A detailed..., released by Goldman Sachs and AWS. AWS Quantum Technologies Blog. Nov. 2023. URL: https://aws.amazon.com/cn/blogs/quantum-computing/a-detailed-end-to-end-assessment-of-a-quantum-algorithm-for-portfolio-optimization-released-by-goldman-sachs-and-aws/.

[3] Fred Glover, Gary Kochenberger, and Yu Du. A Tutorial on Formulating and Using QUBO Models. 2019. arXiv: 1811.11538 [cs.DS]. URL: https://arxiv.org/abs/1811.11538.

[4] Alberto Peruzzo et al. “A variational eigenvalue solver on a photonic quantum processor”. In: Nature Communications 5 (2014), p. 4213. DOI: 10.1038/ncomms5213. URL: https://doi.org/10.1038/ncomms5213.