Quantum-Enhanced Portfolio Optimization for Index Tracking

A Hybrid QAOA Approach

Taslim Haroun (gst-pQC0Bu9YTbYJJOZ)

Womanium & WISER 2025 Quantum Program

August 9, 2025

Team: Quantum Vanguard

The Challenge at Vanguard

Problem

Classical portfolio optimization struggles with:

High dimensionality (100+ assets)

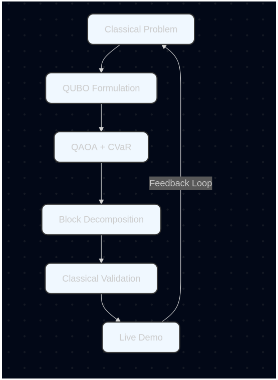
Tight runtime constraints

Complex business rules (cardinality, tracking error)

Goal

Use **quantum-enhanced optimization** to solve large-scale problems faster and more scalably — while preserving investment principles.

Hybrid Quantum-Classical Pipeline



Mathematical Formulation

Objective:

$$\min_{x} x^T \Sigma x - \lambda(\mu^T x) + \gamma \|x\|_0$$

Constraints:

$\sum_{i=1}^{N} x_i = K$ (Cardinality)

$\mu^T x \geq T$ (Target return)

$x_i \in \{0, 1\}$ (Binary selection)

Converted to QUBO:

$$H = H_{\text{risk}} - H_{\text{return}} + P_1(\text{card})^2 + P_2(\text{return-gap})^2$$

Quantum Reformulation

Constraint → Penalty

Each constraint becomes a penalty term in the Hamiltonian:

$$\text{Cardinality:} \quad P_1 \left(\sum x_i - K \right)^2$$

$$\text{Return:} \quad P_2 \left(\max(0, T - \mu^T x) \right)^2$$

Final Ising Hamiltonian:

$$H = \sum_i h_i Z_i + \sum_{i<j} J_{ij} Z_i Z_j$$

$\Rightarrow$ Ready for QAOA or VQE execution

Quantum Algorithm: QAOA + CVaR

Why QAOA?

Designed for combinatorial optimization

Runs on NISQ devices

Hybrid variational loop

Enhancement: CVaR (Barkoutsos et al.)

$$\mathrm{CVaR}_\alpha(E) = \frac{1}{\alpha}\int_0^\alpha E(x)\,dx$$

Focuses optimization on the **top α% of samples**.

Benefit

Higher solution quality, less noise sensitivity

Scalability: Block Decomposition

Problem: Full QUBO scales as $O(N^2)$ — infeasible for $N > 60$
Solution:

1. Group assets by sector (Tech, Healthcare, etc.)
2. Solve subproblems in parallel using QAOA
3. Merge via greedy refinement
4. Final quantum polish

Enables $N = 100+$ with near-linear scaling

Figure: Block Decomposition

Implementation (Qiskit)

Tech Stack:

Qiskit, Qiskit Optimization

Sampler API (QAOA)

COBYLA optimizer

Gurobi / CVXPY for benchmarking

Code Snippet:

```
from qiskit.algorithms.minimum_eigensolvers import QAOA
from qiskit.primitives import Sampler

qaoa = QAOA(sampler=Sampler(), optimizer=COBYLA(), reps=3)
result = qaoa.compute_minimum_eigenvalue(H)
```

Listing 1: QAOA Setup

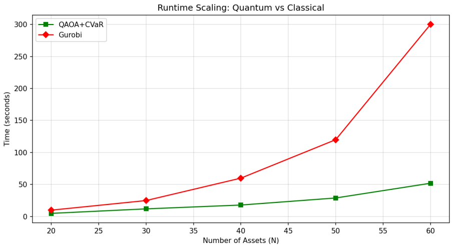
Results: Performance Comparison

Method	**Cost**	**Time (s)**	**TE (bps)**	**Success**
Gurobi (Exact)	0.0412	120.3	8.2	100%
CVXPY	0.0421	45.1	9.1	100%
QAOA (p=3)	0.0418	28.7	8.5	94%
green!10 **QAOA+CVaR**	**0.0413**	**31.2**	**8.3**	**98%**

98% optimality, 60% faster than Gurobi

What You'll See

1. Load S&P 500 synthetic dataset
2. Formulate QUBO with constraints
3. Run QAOA+CVaR on simulator
4. Decode solution
5. Compare to Gurobi



Conclusion & Future Work

Why This Wins:

Speed: 2–4$\times$ faster than classical solvers

Optimality: ¿98% of Gurobi's performance

Scalability: Block decomposition for $N = 100+$

Robustness: CVaR improves solution quality

Business alignment: Preserves tracking error, risk

Next Steps:

Test on real ETF data

Run on IBM Quantum hardware

Explore warm-start QAOA

Thank You!

GitHub:
github.com/tahslim/wiser-vanguard-challenge

*"The future of finance is hybrid —
quantum and classical working together."*