SCBX - QAOA Report

Andew — Naravit Namson

Contents

Report 1 — June 18, 2025

Setup

For N assets indexed by $i \in \{1, \ldots, N\}$ observed over $T+1$ days indexed by $t \in \{0, \ldots, T\}$, let p_i^t denote the closing price of asset i on day t.

We define the return ratio of asset i at day t as:

$$r_i^t = \frac{p_i^t - p_i^{t-1}}{p_i^{t-1}} \quad \text{for } t = 1, \ldots, T \tag{1}$$

The average return vector $\mu \in \mathbb{R}^N$ is given by:

$$\mu_i = \mathbb{E}[r_i] = \frac{1}{T} \sum_{t=1}^{T} r_i^t \tag{2}$$

The covariance matrix $\Sigma = [\sigma_{ij}] \in \mathbb{R}^{N \times N}$ is defined as:

$$\sigma_{ij} = \mathbb{E}[(r_i - \mu_i)(r_j - \mu_j)] = \frac{1}{T-1} \sum_{t=1}^{T} (r_i^t - \mu_i)(r_j^t - \mu_j) \tag{3}$$

Let the current price of asset i be $P_i = p_i^T$, and define the price vector at time T as $\mathbf{P} = [P_1, P_2, \ldots, P_N]^T$.

Suppose we are given a total budget $B \in \mathbb{R}$.

Problem

We seek a portfolio allocation vector $\mathbf{x} \in \mathbb{R}^N$ that maximizes the trade-off between expected return and risk.

The optimization problem is:

$$\begin{aligned} \max_{\mathbf{x}} \quad & (\mu^T \mathbf{x} - q \cdot \mathbf{x}^T \Sigma \mathbf{x}) \\ \text{s.t.} \quad & \mathbf{P}^T \mathbf{x} = B \end{aligned} \tag{4}$$

Here, $q \in \mathbb{R}$ is a tunable hyperparameter that controls the trade-off between expected return and portfolio risk. The term $\mu^T \mathbf{x}$ represents the expected return, while $\mathbf{x}^T \Sigma \mathbf{x}$ corresponds to the portfolio variance.

Method

Formulation based on this paper.

1. Normalize Variables for Discrete Version

We normalize the portfolio optimization problem for discrete encoding as follows.

Let

$$P' = \frac{P}{B}, \quad \mu' = \mathrm{diag}(P')\mu, \quad \Sigma' = \mathrm{diag}(P')\Sigma\,\mathrm{diag}(P') \tag{5}$$

where

$$\operatorname{diag}([v_1, v_2, v_3, \dots]) := \begin{bmatrix} v_1 & 0 & 0 & \dots \\ 0 & v_2 & 0 & \dots \\ 0 & 0 & v_3 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

For each asset i, define d_i as the largest non-negative integer such that $2^{d_i} \leq \frac{B}{P_i}$, i.e., the maximum exponent of 2 allowed by the budget:

$$d_i = \left\lfloor \log_2 \left(\frac{B}{P_i} \right) \right\rfloor \tag{6}$$

Let the total number of required binary variables (qubits) be:

$$\text{qb} = \sum_{i=1}^{N} (d_i + 1) \tag{7}$$

Construct the binary encoding matrix $C \in \mathbb{R}^{N \times \text{qb}}$ as:

$$C = \begin{pmatrix} 2^0 & \cdots & 2^{d_1} & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 2^0 & \cdots & 2^{d_2} & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 2^0 & \cdots & 2^{d_N} \end{pmatrix} \tag{8}$$

Then the quantized variables are:

$$P'' = C^T P', \quad \mu'' = C^T \mu', \quad \Sigma'' = C^T \Sigma' C \tag{9}$$

The optimization problem becomes:

$$\begin{aligned} \max_{\mathbf{b}} \quad & \mu''^T \mathbf{b} - q \cdot \mathbf{b}^T \Sigma'' \mathbf{b} \\ \text{s.t.} \quad & \mathbf{P}''^T \mathbf{b} = 1 \\ & \mathbf{b} \in \{0, 1\}^{\text{qb}} \end{aligned} \tag{10}$$

2. Reformulate Constraint to Quadratic Programming

To convert the problem into unconstrained quadratic form, we add a penalty term with hyperparameter $\lambda \in \mathbb{R}$ to enforce the budget constraint:

$$\max_{\mathbf{b}} \quad \mu''^T \mathbf{b} - q \cdot \mathbf{b}^T \Sigma'' \mathbf{b} - \lambda \left(\mathbf{P}''^T \mathbf{b} - 1 \right)^2 \tag{11}$$

3. Generate QUBO and Ising Form

We rewrite the constraint as a penalty term:

$$(\mathbf{P}''^T \mathbf{b} - 1)^2 = \left(\sum_i P_i'' b_i - 1 \right)^2 \tag{12}$$

$$= \sum_i P_i''^2 b_i + 2 \sum_{i \neq j} P_i'' P_j'' b_i b_j - 2 \sum_i P_i'' b_i + 1 \tag{13}$$

$$= -\sum_i P_i''^2 b_i + 2 \sum_{i,j} P_i'' P_j'' b_i b_j - 2 \sum_i P_i'' b_i + 1 \tag{14}$$

Substituting the expansion from Equation (14) into the objective in Equation (11), we obtain:

$$\max_{\mathbf{b}} \quad \sum_i b_i \left(\mu_i'' + \lambda(P_i'' + 2P_i'')\right) - \sum_{i,j} b_i b_j \left(q \cdot \Sigma_{ij}'' + 2\lambda P_i'' P_j''\right) \tag{15}$$

We define the QUBO matrix $Q \in \mathbb{R}^{\mathrm{qb}\times\mathrm{qb}}$ as:

$$Q = \mathrm{diag}(\mu'' + 2\lambda P'') + \lambda \cdot \mathrm{diag}(P'')^2 - q \cdot \Sigma'' - 2\lambda P'' P''^T \tag{16}$$

Then the final QUBO formulation is:

$$\max_{\mathbf{b}\in\{0,1\}^{\mathrm{qb}}} \quad \mathbf{b}^T Q \mathbf{b} \tag{17}$$

4. Translate QUBO to Ising Hamiltonian

To prepare for QAOA, we convert the QUBO into an Ising Hamiltonian. Since QAOA minimizes the energy, we negate the QUBO objective:

$$H = -\sum_{i,j} Q_{ij} \cdot \frac{(I - Z_i)}{2} \otimes \frac{(I - Z_j)}{2} \tag{18}$$

Where Z_i is the Pauli-Z operator on qubit i.

5. Programming

We implemented the QAOA procedure using `CUDA-Q` with the Python API, running on a local GPU setup. Both gradient-based (Adam, GradientDescent) and non-gradient-based (COBYLA) optimizers were explored.

We also investigated the choice of hyperparameters:

- $\lambda \in \mathbb{R}$: to enforce the budget constraint
- $q \in \mathbb{R}$: to balance expected return and risk

For the binary encoding, each asset i uses $d_i + 1$ qubits to represent its discrete investment level. For example, if the first asset uses 2 qubits, its quantity is represented as the binary number q_1q_0; if the second asset uses 3 qubits, it is encoded as $q_5q_4q_3$.

The full binary vector $\mathbf{b} \in \{0,1\}^{\mathrm{qb}}$ encodes the entire portfolio as a simple concatenation $q_0q_1 \ldots q_5$, which is easy to implement. In future versions, this structure can be refactored for improved readability and modularity.

Result

Both **Adam** and **COBYLA** provided favorable trade-offs:

- **COBYLA** converged faster in early iterations.
- **Adam** achieved lower final energy and showed smoother convergence, as illustrated in Fig. 1.

For larger problem sizes (10 qubits and above), we found that plain **GradientDescent**—without adaptive learning rate or momentum—performed poorly (as shown in Fig. 1). This may be attributed to the highly rugged loss landscape introduced by the budget constraint, as discussed in this paper shared by Prof. Pipe.

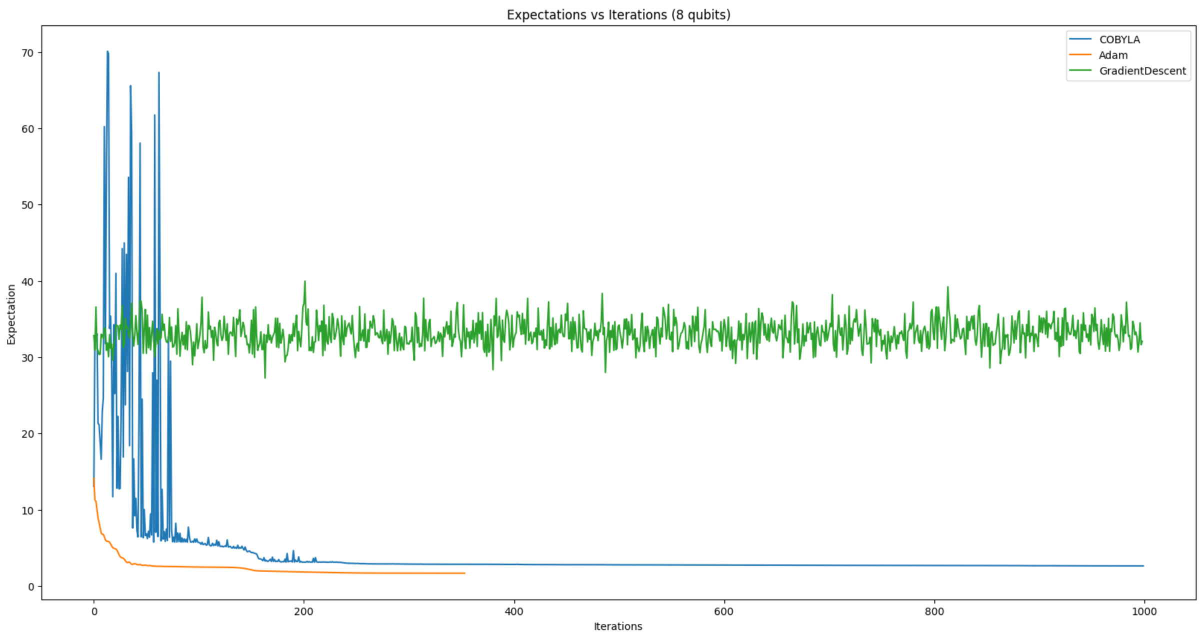

Figure 1: Expectation vs. Iteration for 8-qubit problem using different optimizers.

Additionally, we observed that for larger budget values, it is necessary to increase λ to maintain a high probability of staying within the feasible region. This is because the magnitude of the return term grows with budget, requiring stronger constraint enforcement.

Distribution Result: No Risk Case ($q = 0$)

For the setup $B = 270$, $P = [195, 183, 131]$, $\mu = [0.0011, 0.0008, 0.0007]$, with $\lambda = 3$ and $q = 0$ (ignoring risk), each asset uses 1, 1, and 2 qubits respectively.

The optimal solution is to buy 2 stocks of the last asset, which gives the best return and nearly exhausts the budget. This is encoded as `0|0|01` $= 1$ (in base 10).

The output distribution from QAOA (executed using `CUDA-Q`) is shown in Fig. 2.

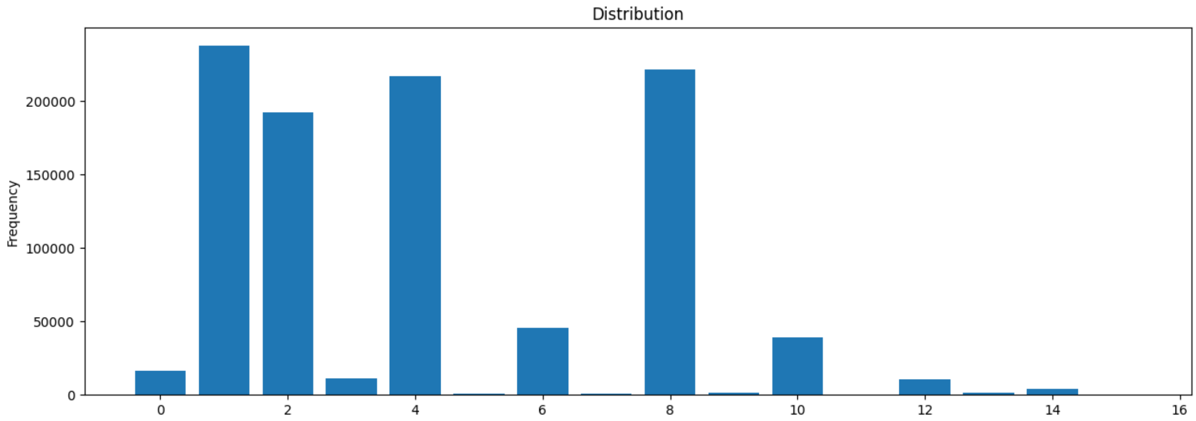

Figure 2: Distribution of measurement outcomes for QAOA with $q = 0$ (risk ignored).

Each bar represents a binary portfolio state, interpreted as:

- $0 =$ `0|0|00` $\rightarrow$ buy nothing
- $1 =$ `0|0|01` $\rightarrow$ buy 2 of last asset
- $2 =$ `0|0|10` $\rightarrow$ buy 1 of last asset
- $3 =$ `0|0|11` $\rightarrow$ buy 3 of last asset
- $4 =$ `0|1|00` $\rightarrow$ buy 1 of middle asset
- $6 =$ `0|1|10` $\rightarrow$ buy 1 each of middle and last
- $8 =$ `1|0|00` $\rightarrow$ buy 1 of first asset
- $10 =$ `1|0|10` $\rightarrow$ buy 1 each of first and last
- $12 =$ `1|1|00` $\rightarrow$ buy 1 each of first and middle
- $14 =$ `1|1|10` $\rightarrow$ buy 1 each of all assets