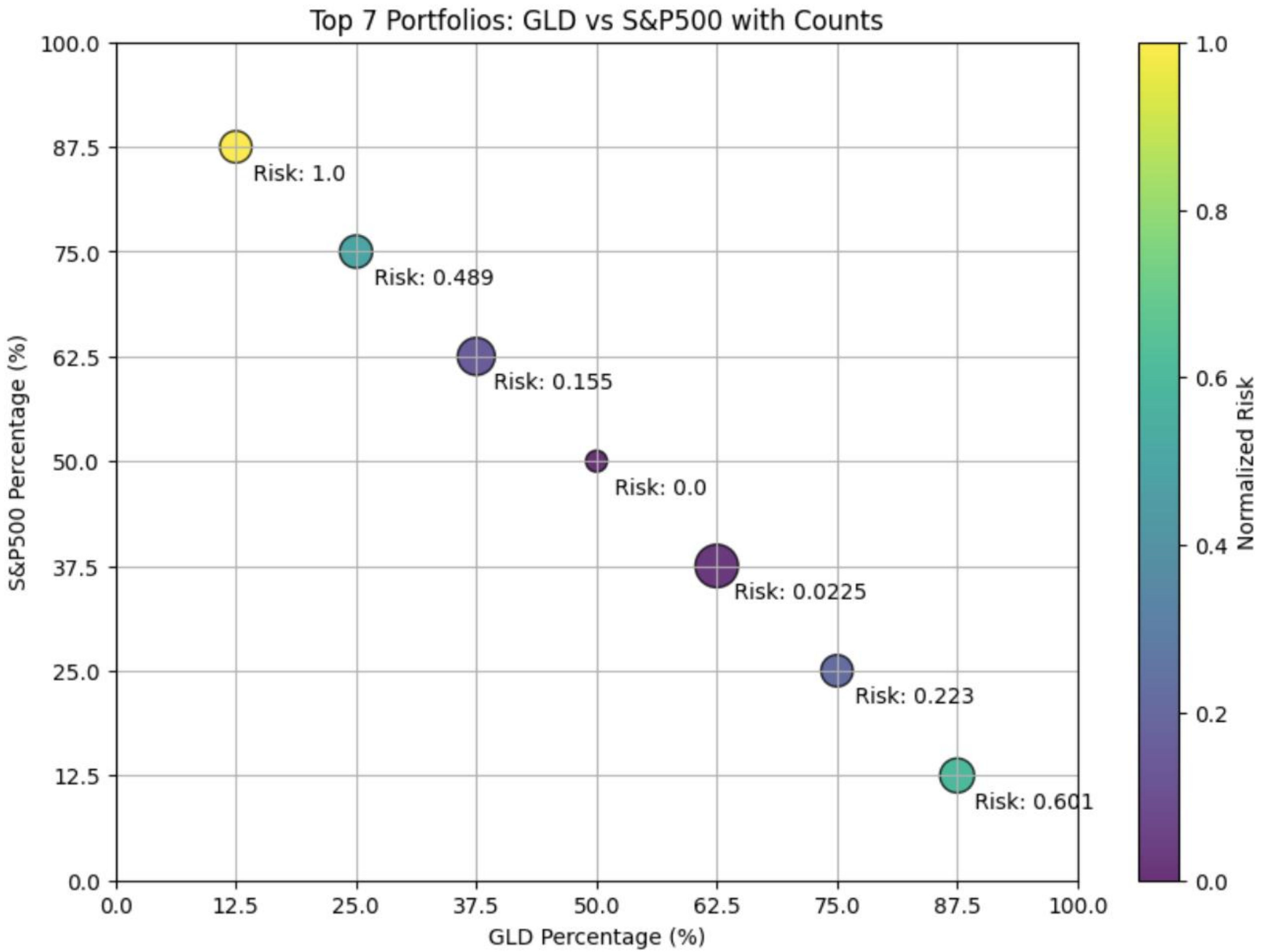
# QAOA for portfolio optimization

Edvardas Ražanskas, ISI

What is portoflio optimization?



# The Classical Approach to Portfolio Optimization

- Markowitz Mean-Variance Model (Modern Portfolio Theory)
- Goal: find portfolio with minimal risk(variance) and maximum return, subject to constraint (budget can't exceed 100%)
- Limitations: Computationally intensive for large portfolios (many different assets).

# The Quantum Approach to Portfolio Optimization

- Encoding the portfolio optimization problem as a Quadratic Unconstrained Binary Optimization (QUBO) problem.
- Goal: find portfolio with minimal risk(variance) and maximum return, subject to constraint (budget can’t exceed 100%

# The Quantum Approach to Portfolio Optimization

QAOA for portfolio optimization:

1. Data Preparation: Get and prepare portfolio (e.g. stock) data;
2. Encoding the portfolio optimization problem as QUBO (Quadratic Unconstrained Binary Optimization);
3. Encode the QUBO as a quantum Hamiltonian;
4. Construct a parameterized quantum circuit (QAOA circuit);
5. (Use classical optimization to tune circuit parameters for optimal solution)

Output: Probabilities of different portfolio allocations.

# Data Preparation

- Data source: Historical price data for selected assets (e.g., GLD, S&P 500);

```
daily_returns.head()
```

| Ticker | GLD | ^GSPC |
|---|---|---|
| Date | | |
| 2005-01-04 | -0.650857 | -1.167136 |
| 2005-01-05 | -0.163789 | -0.362784 |
| 2005-01-06 | -1.218647 | 0.350586 |
| 2005-01-07 | -0.735472 | -0.143117 |
| 2005-01-10 | 0.262908 | 0.342277 |

# Calculate daily mean returns and covariance matrix

```
cov_stocks=daily_returns.cov()
cov_stocks
```

| Ticker | GLD | ^GSPC |
|---|---|---|
| **Ticker** | | |
| **GLD** | 1.221527 | 0.076211 |
| **^GSPC** | 0.076211 | 1.458465 |

```
returns_list = daily_returns.mean().to_numpy().tolist()
returns_list
```

```
[0.04045768843899977, 0.03887037743039809]
```

# QUBO Construction

```
# Construct the QUBO matrix
Q = construct_portfolio_qubo(cov_matrix, n_assets, bits_per_asset,
                             budget, returns_list, 2, lambd)
print(Q)
```

```
[[-2.33477067  0.3506875   0.701375    0.1574375   0.314875    0.62975   ]
 [ 0.3506875  -4.31885384  1.40275     0.314875    0.62975     1.2595    ]
 [ 0.701375    1.40275    -7.23495769  0.62975     1.2595      2.519     ]
 [ 0.1574375   0.314875    0.62975    -2.33068634  0.3580625   0.716125  ]
 [ 0.314875    0.62975     1.2595      0.3580625  -4.30331019  1.43225   ]
 [ 0.62975     1.2595      2.519       0.716125    1.43225    -7.17437038]]
```

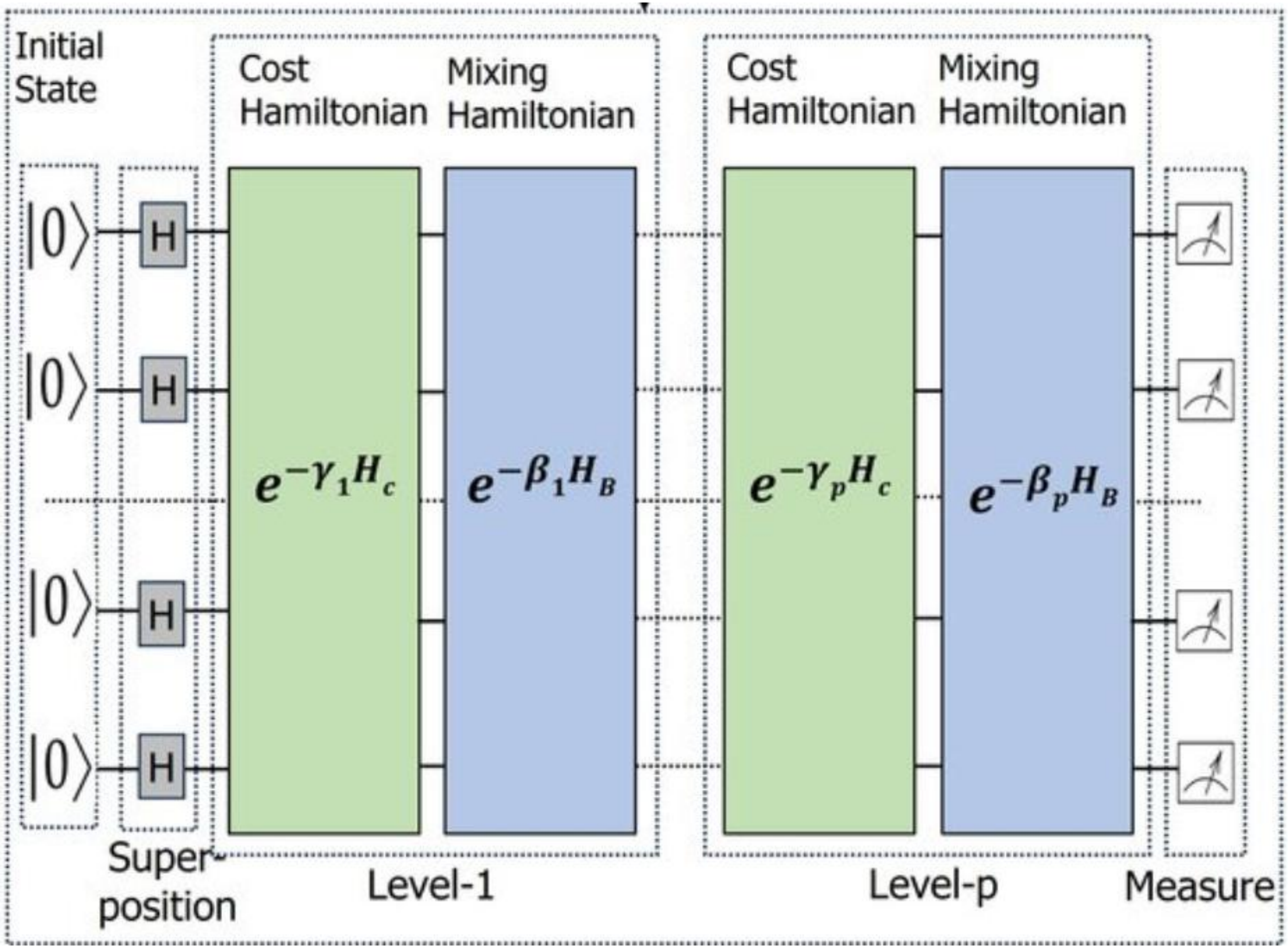
# Encode the QUBO as a quantum Hamiltonian;

```
# Encode the QUBO as a quantum Hamiltonian
h, J, offset = from_Q_to_Ising(Q, offset)

# QAOA parameters
p = 5  # Number of QAOA layers
gammas = np.linspace(0.1, 1.0, p)  # Parameters for cost Hamiltonian
betas = np.linspace(0.1, 0.8, p)[::-1]  # Parameters for mixer Hamiltonian
total_qubits = n_assets * bits_per_asset

# Create the QAOA circuit
qc = qaoa_circuit(gammas, betas, h, J, total_qubits)
```

# Quantum Circuit Implementation

- Initial state: Superposition of all possible allocations.
- Alternating layers: Cost Hamiltonian (risk/return), Mixer Hamiltonian (exploration).
- Measurement: Extract bitstrings representing portfolio allocations.

# Quantum Circuit Implementation



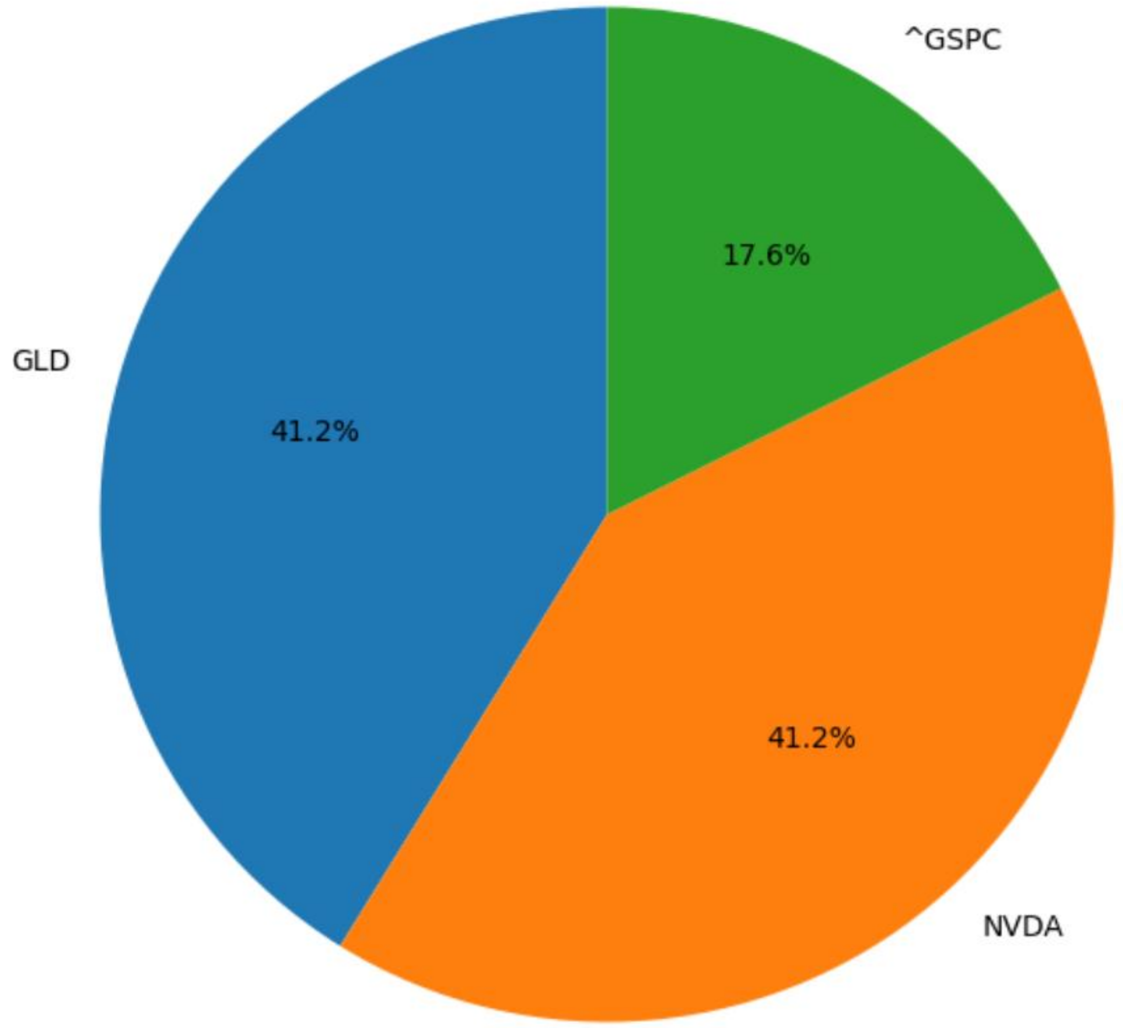
# Post-Processing and Portfolio Evaluation

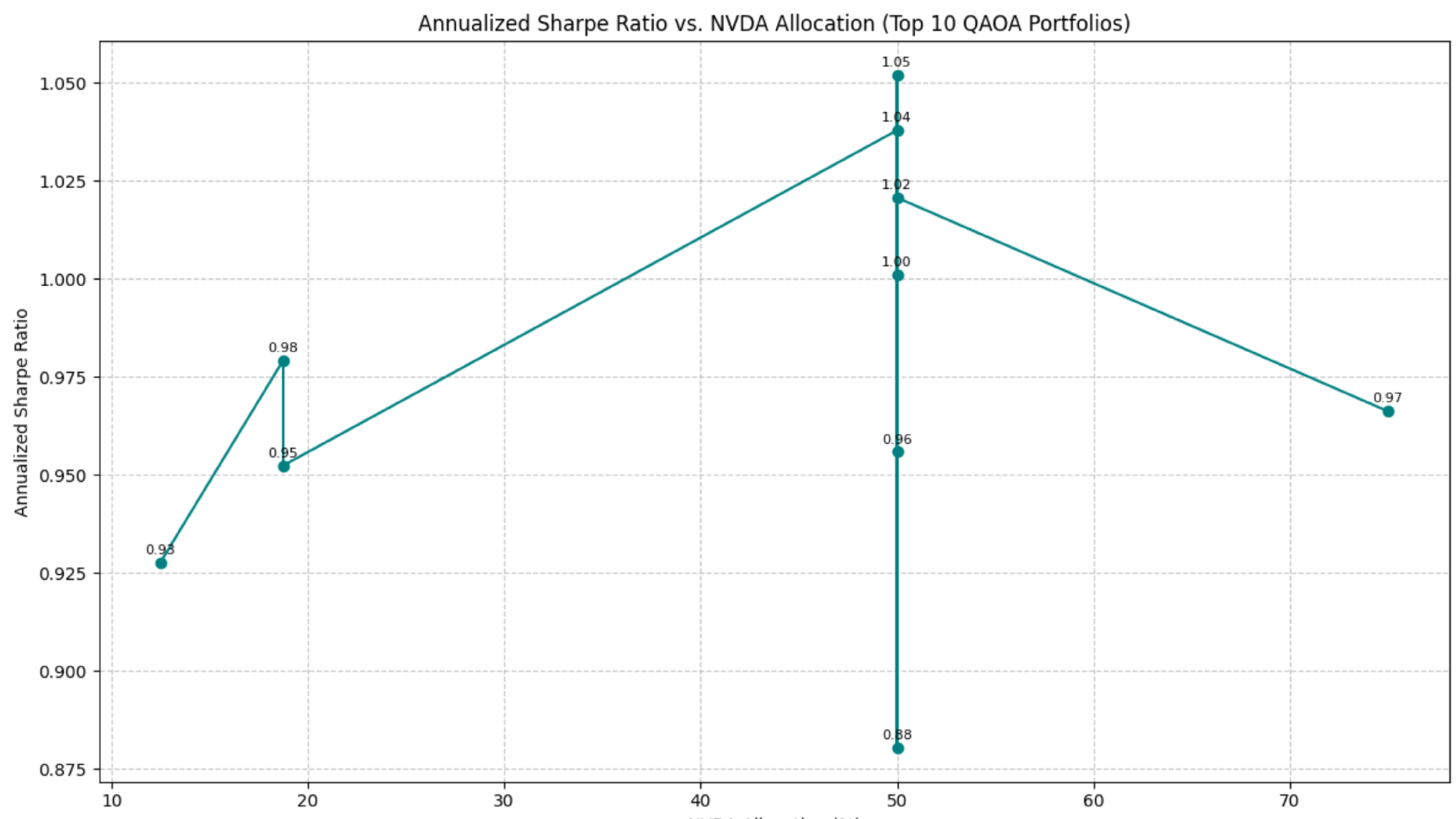
- Decode quantum measurement results into asset allocations.
- Calculate portfolio risk (variance) and expected return for each allocation.
- Filter valid portfolios (satisfying constraints).
- Compute Sharpe ratio for risk-adjusted performance.

# Results with 300’000 shots (top 3 portfolios shown)

```
 1
Count: 70, bitstring: 001101110111, allocations: {0: 0.4375, 1: 0.4375, 2: 0.1875}
GLD: 43.75%
NVDA: 43.75%
^GSPC: 18.75%

 2
Count: 55, bitstring: 011100110111, allocations: {0: 0.4375, 1: 0.1875, 2: 0.4375}
GLD: 43.75%
NVDA: 18.75%
^GSPC: 43.75%

 3
Count: 54, bitstring: 011101110011, allocations: {0: 0.1875, 1: 0.4375, 2: 0.4375}
GLD: 18.75%
NVDA: 43.75%
^GSPC: 43.75%
```

Optimal Portfolio Allocation



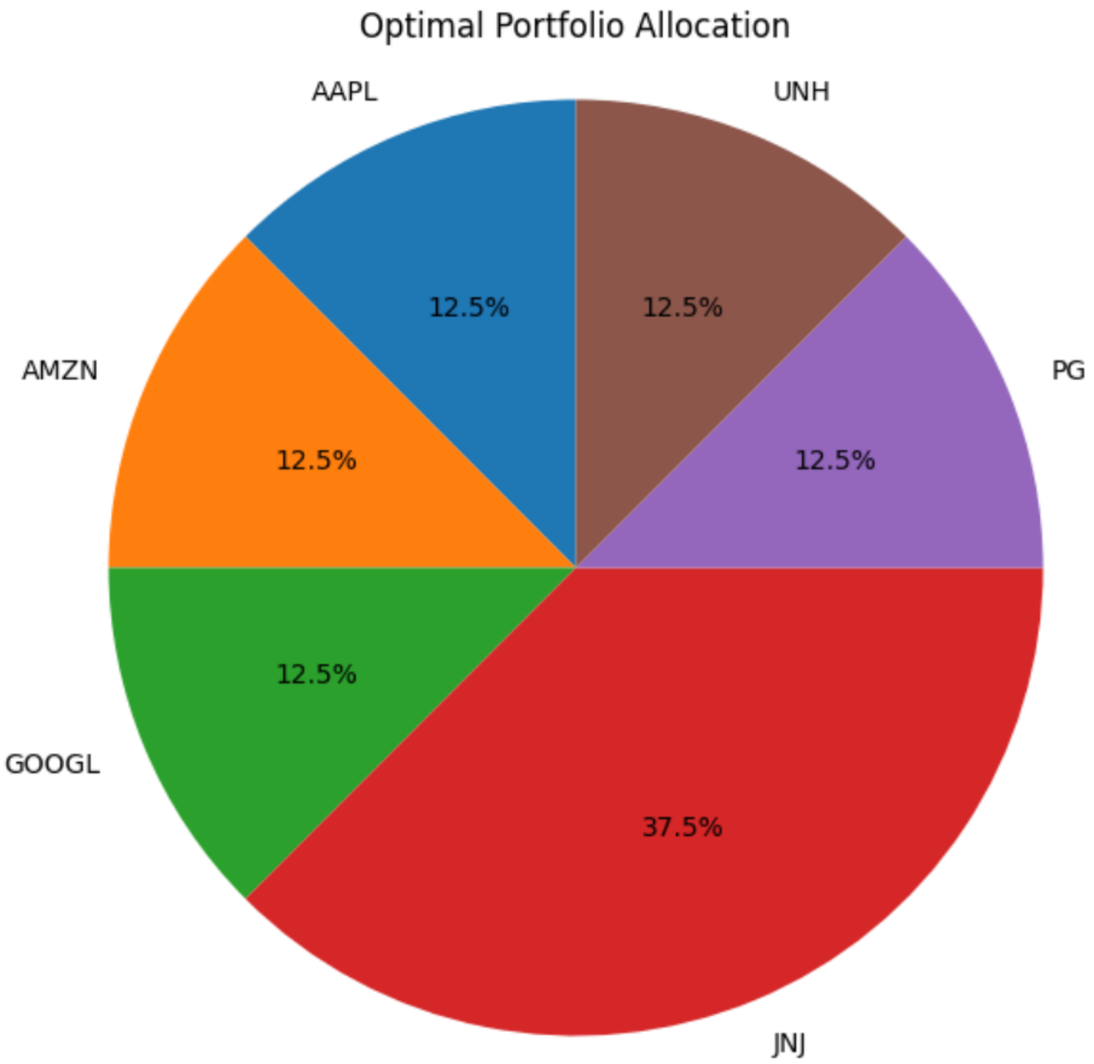


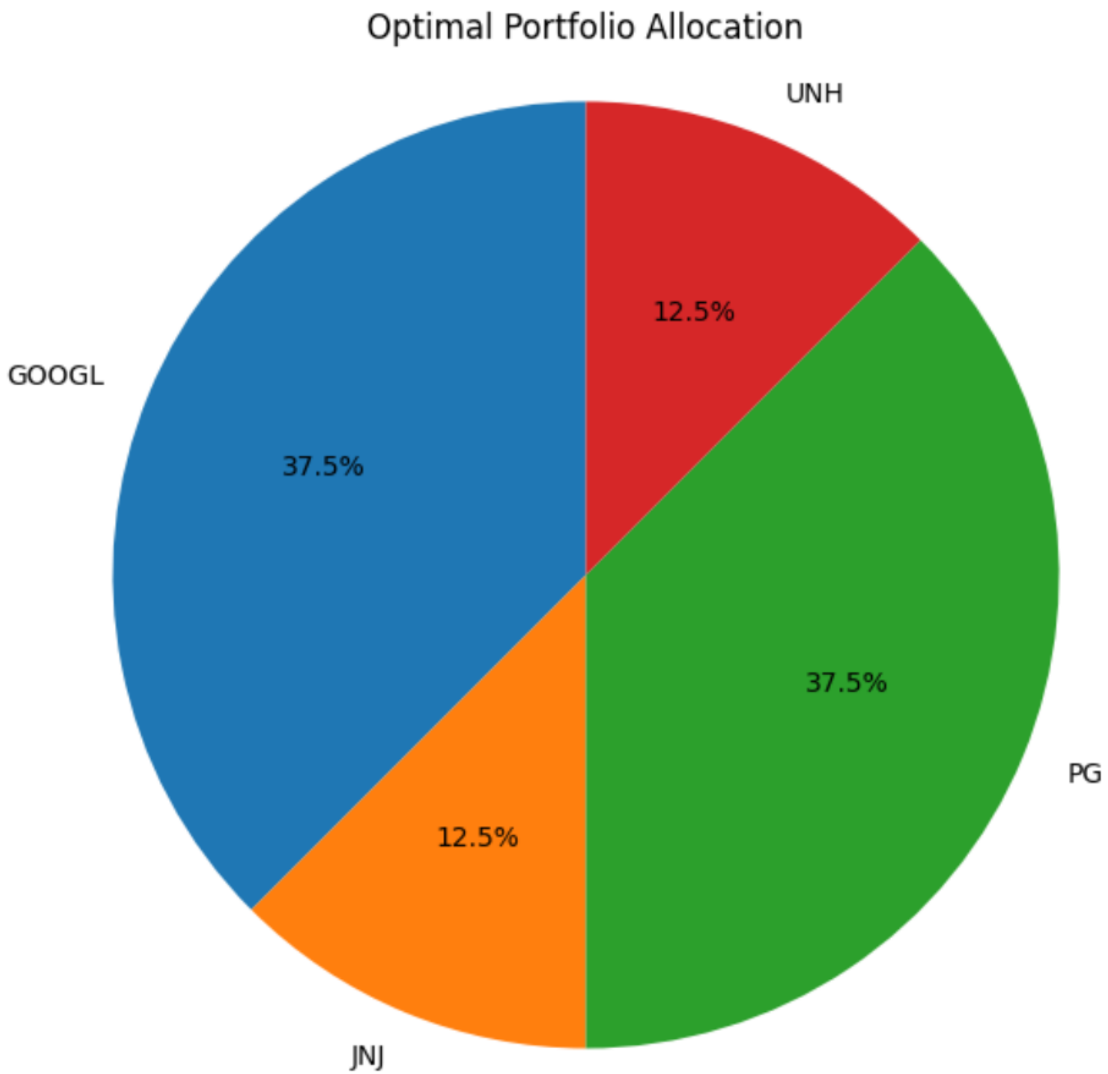
Annualized Sharpe Ratio vs. NVDA Allocation (Top 10 QAOA Portfolios)
Annualized Sharpe Ratio
1.050
1.025
1.000
0.975
0.950
0.925
0.900
0.875
10
20
30
40
50
60
70
1.05
1.04
1.02
1.00
0.98
0.95
0.93
0.97
0.96
0.88

# For 10 stocks with 8 bit accuracy not enough memory

```
[*********************100%***********************]  10 of 10 completed
---------------------------------------------------------------------------
QiskitError                               Traceback (most recent call last)
Cell In[86], line 40
     38 simulator = AerSimulator()
     39 result = simulator.run(qc, shots=shots_number).result()
---> 40 result_counts = result.get_counts()

File D:\Documents\anaconda3\envs\quant\Lib\site-packages\qiskit\result\result.py:264, in Result.get_counts(self, experiment)
    262 dict_list = []
    263 for key in exp_keys:
--> 264     exp = self._get_experiment(key)
    265     try:
    266         header = exp.header.to_dict()

File D:\Documents\anaconda3\envs\quant\Lib\site-packages\qiskit\result\result.py:392, in Result._get_experiment(self, key)
    390 result_status = getattr(self, "status", "Result was not successful")
    391 exp_status = getattr(exp, "status", "Experiment was not successful")
--> 392 raise QiskitError(result_status, ", ", exp_status)

QiskitError: 'ERROR:  [Experiment 0] Insufficient memory to run circuit circuit-173 using the statevector simulator. Required memory: 16384M, max memory: 1
6074M ,  ERROR: Insufficient memory to run circuit circuit-173 using the statevector simulator. Required memory: 16384M, max memory: 16074M'
```

# 6 stocks allocation with 3 bit accuracy



# Conclusions and Future Directions

- QAOA successfully solves small-scale portfolio optimization problems (e.g., 3–10 assets) with results matching classical benchmarks;
- Theoretical speedup potential for large portfolios with quantum parallelism;
- Hardware limitations: Current NISQ devices restrict portfolio size.
- Future work: Layerwise optimization, error mitigation, and hybrid quantum-classical solvers .