Currency Arbitrage Optimization using QAOA in Qiskit and Comparative Study with Classical Baselines

Authors:

Saranya Sundararajan (2577421)
Bhuvanesh Kumar (1844341)
Santhossh J U (1982102)
Mohammad Yaseen (2160066)
Gowthaman A (1969311)
Date: January 2026

Abstract

This project formulates currency arbitrage detection as a Quadratic Unconstrained Binary Optimization (QUBO) problem and solves it using the Quantum Approximate Optimization Algorithm (QAOA) implemented in Qiskit. Exchange rates are provided as a square matrix and treated as a directed weighted graph (row currency -> column currency). A profitable arbitrage corresponds to a directed cycle whose product of exchange rates is greater than 1. To make the objective QUBO-friendly, the multiplicative product of rates is converted to an additive objective using the transformation -log(rate). We then build a position-based (N, L) fixed cycle-length QUBO (N = number of currencies, L = trading cycle length), convert it to an Ising Hamiltonian, and apply a QAOA workflow that matches the team lead's procedure: start in superposition, apply cost and mixer layers, sample, tune with a classical optimizer, and pick a winner from feasible samples. The implementation includes feasibility validation (one currency per position, no repeated currencies, and no invalid zero-rate transitions), classical brute-force baselines for small instances, and fee sensitivity analysis.

1. Introduction

Currency arbitrage exploits inconsistencies in exchange rates to generate profit through cyclic trades such as: EUR -> USD -> CAD -> EUR. If the product of the exchange rates along the cycle exceeds one, the cycle is profitable. A directed graph representation is natural: Nodes are currencies. A directed edge i -> j exists when an exchange rate R[i,j] > 0 is available. The edge weight is the exchange rate.

2. Motivation

Classical search becomes expensive as the number of currencies and cycle length grows. QAOA provides a quantum-classical hybrid approach that explores a large solution space using parameterized quantum circuits, while a classical optimizer tunes the circuit parameters. The goal of this work is not to claim quantum advantage at current scales, but to provide: A correct QUBO/Ising model of fixed-length currency cycles, A robust QAOA simulation pipeline in Qiskit, Classical baselines and diagnostics to validate correctness.

3. Dataset (matrix format)

Input data is stored in: data/Data-sheet-New.csv (and data/Data-sheet-New.txt). Format: The first row and first column contain currency labels. The entry R[i,j] is the exchange rate from row currency i to column currency j. R[i,j] = 0 indicates that a direct conversion is not available and is treated as an invalid transition. Default dataset size: N = 14 currencies.

4. Problem Statement and Formulation (position-based fixed cycle length)

We solve for a trading cycle of fixed length L (where 2 <= L <= N). Decision variables (position-based encoding): For each position p in {0, ..., L-1} and currency index i in {0, ..., N-1}: x[p,i] in {0,1}. Objective: minimize sum over p,i,j of (-log(R[i,j])) x[p,i] x[(p+1) mod L, j]. Constraints: one currency per position, no repeated currencies, valid transitions only.

5. Methodology: QAOA in Qiskit (team lead’s 6 steps)

1) Start Everywhere (superposition) 2) Twist for Profit (cost layer) 3) Shake it Up (mixer layer) 4) Check Signal (sampling) 5) Tune Again (classical optimization) 6) Pick Winner (decode, filter, report). Implementation uses Qiskit 2.3.0, qiskit-aer 0.17.2, qiskit-optimization 0.7.0, qiskit-algorithms 0.4.0.

6. Experimental Details (default configuration)

Default parameters: N = 14, L = 3, num_variables = 42. QAOA: reps = 2, shots = 2048, maxiter = 60. Penalty weights: position = 50.0, repeat = 50.0, invalid_edge = 200.0.

7. Results and Discussion

Classical brute-force baseline (L=3): Best cycle: EUR -> USD -> HUF -> EUR. Gross return: 1.0009526828. Profit: 0.095268%. Break-even fee per trade: ≈ 0.031736%. QAOA output includes most profitable and most probable feasible cycles.

8. Comparison and Analysis

Classical brute-force is exact but scales poorly. QAOA embeds the cycle search in a Hamiltonian and explores via quantum sampling. Feasibility constraints are enforced via QUBO penalties.

9. Runtime and Scaling Notes

QAOA execution is time-consuming due to circuit evaluations in optimizer loop. ETA is printed during optimization. To reduce runtime: lower maxiter, reps, shots, or reduce N/L.

10. Conclusion

This project demonstrates that currency arbitrage can be modeled as a position-based fixed-length cycle QUBO and solved using QAOA in Qiskit. The notebook is modular, robust, and validated with classical baselines.

Appendix: Key Code Snippets (high level)

Data file discovery: upward search. QUBO variable mapping: x[p,i] for positions p and currencies i. Cycle decoding: one currency per position, no repeats, valid transitions only.