Optimal Currency Arbitrage Opportunity Detection

Problem Statement

The optimal currency arbitrage opportunity detection problem is a classic problem in computational finance and algorithmic trading. It focuses on detecting whether a set of currencies and exchange rates allows a trader to start with one currency, exchange through a sequence of others, and end up with more of the original currency - without risk and without net investment.

Even small opportunities (0.01%-0.1%) can be highly valuable when executed at scale or in high-frequency environments. However, detecting such opportunities in real time is extremely challenging due to the combinatorial explosion of possible currency conversion paths.

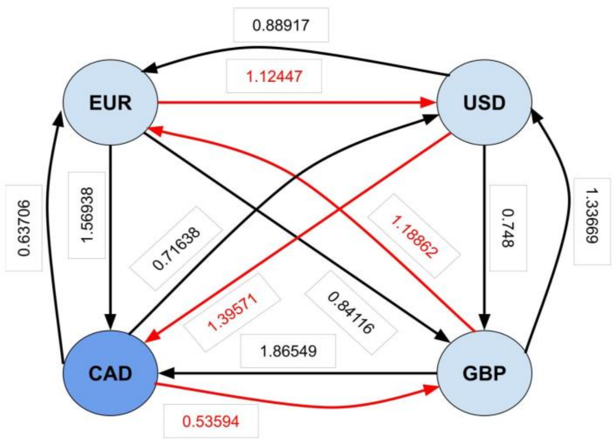
In this challenge, participants are expected to develop a quantum or quantum-classical solution that addresses the problem with the provided data and uncovers the most profitable currency-arbitrage opportunity.

Example:

In global currency markets, exchange rates between currencies (USD, EUR, INR, JPY, etc.) are always changing.

CAD > GBP> EUR > USD > CAD

If the conversion rates multiply to more than 1, you make a risk-free profit, known as **Currency Arbitrage**.



Pictorial view of an example of expected output.

Sample Mathematical Model

Objective: Maximize the total profit.

Constraints:

1. At each position of the trading cycle, exactly one currency should be assigned.
2. To complete the trading cycle, the first and last positions should be assigned to the same currency.
3. No sub-loop is allowed in the resulting trading cycle.

4. Two non-converted currencies (currencies which are not directly converted to each other) cannot be assigned in the adjacent positions of the resultant trading cycle.

Current Challenges

In this problem, there are hundreds of currencies, thousands of exchange-rate paths; prices change every millisecond, and the number of possible conversion cycles becomes too large for classical computers to check quickly. Most profitable arbitrage opportunities last only milliseconds, so time is everything.

Why is this a good problem for exploring a Quantum solution?

- To find an arbitrage, you must search through a huge number of possible currency paths. Quantum computers can explore many possibilities at once (superposition), giving them a natural advantage.

- Arbitrage detection can be written as an optimization problem to find the best (most profitable) cycle. Quantum techniques like QAOA or quantum annealing are designed specifically for this kind of problem, possibly giving faster results.

- Quantum methods can reduce computation time, search complexity, and missed opportunities. This makes it attractive for high-frequency trading, where even small speed gains can mean real money.

Expected Output

The expected output is the optimized currency arbitrage trading cycle which will provide maximum profit and ensure all the constraints mentioned.

Evaluation metric: Profit**,** quantum resource (number of qubits, circuit depth, etc.) utilization.

Evaluation Criteria

For first phase of the challenge participants should provide a approximate required quantum resouces (qubit counts, circuit depth, etc) to execute their approach in real quantum hardware.

Background work done: Quality of background work done to understand the use case and the state of the art.

Innovation Quotient: The overall innovation quotient in terms of originality and novelty seen in the approach, concept, and algorithm.

Comprehensiveness: Level of detail, coverage of the tasks towards the targeted goal of the challenge.

Technical completeness: Quality and completeness of the code/solution and its ability to execute and produce results as documented.

Comparison with internal benchmarks: How well the solution compares with any benchmark results that may have been achieved by the organizers.

Extensibility: Ease of adapting/modifying the solution to address variants of the use case (especially additional complexities).

Resource requirements & Scalability: Any resource estimation provided for the solution to indicate the potential future scaling of the solution.

Pitch: Clarity, demonstration, and structure of the overall pitch.

Input Data

Currency exchange rate among 14 currencies for a time instance. Please use this data to solve the challenge. From left to right in the table, the entries are the bid price across currencies. Currencies which are not converted to each other directly 0 are taken as bid price.

	EUR	USD	GBP	CAD	CHF	JPY	AUD	CZK	HUF	NZD	SEK	SGD	DKK	NOK
EUR	1	1.12447	0.84116	1.56938	0.938	162.844	1.74452	24.8693	401.986	1.89895	10.8786	1.45587	7.45944	11.5808
USD	0.88917	1	0.748	1.39571	0.83412	144.806	1.55125	22.1154	357.492	1.68922	9.67615	1.29474	6.63365	10.30045
GBP	1.18862	1.33669	1	1.86549	1.11501	193.575	2.07385	0	0	2.25746	0	1.7305	0	0
CAD	0.63706	0.71638	0.53594	1	0.59761	103.757	1.11154	0	0	1.21004	0	0.92755	0	0
CHF	1.06587	1.19868	0.89659	1.6726	1	173.576	0	0	0	0	0	0	0	0
JPY	0.00614	0.0069	0.00517	0.00964	0.00576	1	0.01071	0	0	0.01166	0	0.00894	0	0
AUD	0.57312	0.64451	0.48207	0.89936	0	93.328	1	0	0	1.08844	0	0.83431	0	0
CZK	0.0402	0.0452	0	0	0	0	0	1	0	0	0	0	0	0
HUF	0.00249	0.00279	0	0	0	0	0	0	1	0	0	0	0	0
NZD	0.52638	0.59183	0.44279	0.82607	0	85.717	0.91852	0	0	1	0	0.76629	0	0
SEK	0.09187	0.1033	0	0	0	0	0	0	0	0	1	0	0	0
SGD	0.68669	0.77222	0.5777	1.07776	0	111.842	1.19816	0	0	1.30439	0	1	0	0
DKK	0.13403	0.15071	0	0	0	0	0	0	0	0	0	0	1	0
NOK	0.0863	0.09704	0	0	0	0	0	0	0	0	0	0	0	1

Sample Pseudo code:

```
def main_function (input_dataframe, number_of_currencies, trading_cycle_length):
    '''
    input_dataframe:<DataFrame> Pandas dataframe of the given given data
    number_of_currencies: <int> : With how many currencies you want to work.
    trading_cycle_length: <int> <= number_of_currencies : Number of trades in your solution.
    '''
    # Creation of the mathematical model of the given problem using given data
    def mathematical_model_creation (input_dataframe, number_of_currencies, trading_cycle_length):
        # write your code here
        return math_model
    # Convert the mathematical model into a quantum suitable form
    def create_quantum_suitable_form( math_model):
        # write your code here
        return quantum_model
    # Quantum algorithm / quantum-classical hybrid algorithm to find the min/max of the quantum model
    def quantum_algo(quantum_model):
        # write your code here
        return binary_solution
    # Post process the solution getting from quantum_algo into a human understandable form
    # example of trading_cycle :  EUR >> USD>> CAD >> EUR
    def post_processig(binary_solution):
        # write your code here
        return profit, trading_cycle

    mdl = mathematical_model_creation(input_dataframe, number_of_currencies, trading_cycle_length)
    q_mdl = create_quantum_suitable_form(mdl)
    solution = quantum_algo(q_mdl)
    profit, trading_cycle = post_processig(solution)
    return profit, trading_cycle

input_dataframe =  "read the given data"
number_of_currencies = <int>
trading_cycle_length = <int> #( less than or equal to number_of_currencies)

profit, trading_cycle = main_function(input_dataframe, number_of_currencies, trading_cycle_length)
```

References

1. https://arxiv.org/abs/2509.09289
2. https://www.tandfonline.com/doi/abs/10.1080/14697688.2019.1639801
3. https://1qbit.com/files/white-papers/1QBit-White-Paper-%E2%80%93-Finding-Optimal-Arbitrage-Opportunities-Using-a-Quantum-Annealer.pdf