DEEP DIVE

The Quest for Optimal Arbitrage Opportunities

Aman Bilkhoo
Head of Quantitative Trading
King's Capital

November 2023

The Quest for Optimal Arbitrage Opportunities

Aman Bilkhoo - Head of Quantitative Trading - King's Capital - Nov 2023

Introduction to Arbitrage

A textbook definition of arbitrage is "the simultaneous purchase and sale of the same, or essentially similar, security in two different markets for advantageously different prices" (Sharpe and Alexander 1990).

This can manifest in many forms and one example is to exploit a difference in pricing of the same stock at two exchanges: simultaneously buy at the low price and sell at the high price to generate a profit. Cross-currency arbitrage, also called circular arbitrage, is another variant and it involves exploiting pricing discrepancies amongst currencies in a foreign exchange market, in which converting between currencies in a cycle (e.g. GBP→EUR→CHF→GBP) results in a profit.

In theory, this type of trading carries zero risk since it does not involve predicting the market and it is known in advance that a profitable arbitrage opportunity exists, before it is exploited. In practice, risks such as execution risk is entailed since it may not be possible to simultaneously execute the necessary trades to exploit the arbitrage opportunity and the trades may take time to complete. During this time, the pricing of the assets involved may change such that it not possible to profit from continuing to trade the opportunity. Furthermore, other market frictions such as transaction costs and taxes also impact the feasibility of profiting from arbitrage opportunities [1].

Arbitrage opportunities are often very short-lived since arbitrageurs would trade to profit from an identified opportunity, continuing to buy the relatively underpriced assets involved (creating excess supply) and continuing to sell the relatively overpriced assets involved (creating excess demand) until the mispricing is driven to correction and the arbitrage opportunity no longer exists [2]. This is a reason why many financial models and pricing calculations assume the absence of arbitrage opportunities. However, for our purposes, the short-term nature of these opportunities motivates the development of efficient computational techniques to find and exploit them quickly to beat the competition.

Initial graph-theoretical specification of cross-currency arbitrage

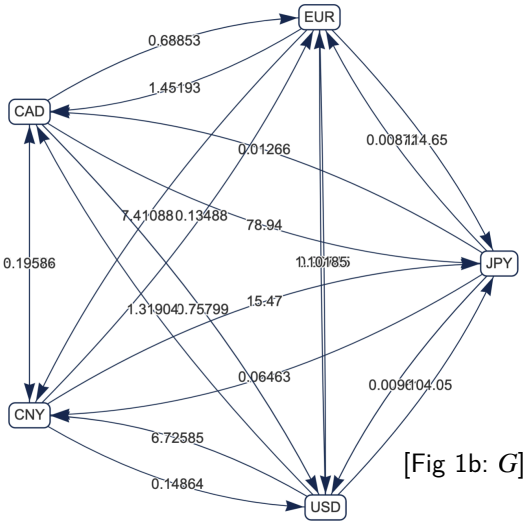
Given a set of currencies and exchange rates between them, the foreign exchange market can be defined as directed weighted graph G, where each currency $c_1, c_2, \ldots, c_n \in C$ is represented with a node and each edge (c_i, c_j) exists if it is possible to convert c_i to c_j and is assigned weight W_{ij}, defined as the best ask price for this trade, possibly accounting for per-unit transaction costs.

An example of a foreign exchange market represented in this way is shown below.

$C = \{\textbf{JPY}, \textbf{EUR}, \textbf{CAD}, \textbf{CNY}, \textbf{USD}\}$

	JPY	EUR	CAD	CNY	USD
JPY	1.0	0.00872	0.01266	0.06463	0.00961
EUR	114.65	1.0	1.45193	7.41088	1.10185
CAD	78.94	0.68853	1.0	5.10327	0.75799
CNY	15.47	0.13488	0.19586	1.0	0.14864
USD	104.05	0.90745	1.31904	6.72585	1.0

[Fig 1a: Adjacency matrix of G. Source of data: Rosenberg [9]]



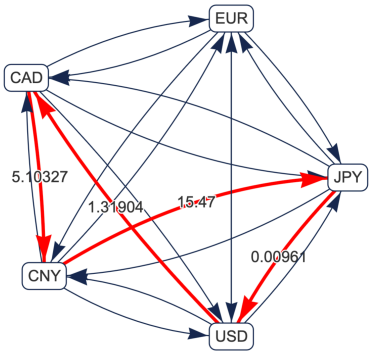
[Fig 1b: G]

Properties that can be observed from this graph include [4]:

- $W_{ii} = 1$ [A rate of 1 is assigned to converting from one currency to itself]
- $W_{ij} > 0$ [Exchange rate ask price from currency c_i to c_j] is always positive]
- $W_{ij} \approx \frac{1}{W_{ji}}$ [The forward and backward exchange rate are approximately equivalent, differences are typically within bid-ask spreads]
- This graph is complete [any currency can be converted to any other currency]

Consider starting with one USD, then trading it between other currencies, then back to USD in this market. If an arbitrage opportunity exists, exploiting it would result in more than one USD after trading in this manner.

In our example, most trades of this type, such as USD→JPY→CAD→USD, result in a loss. However, arbitrage opportunities exist, such as USD→CAD→CNY→USD which results in a profit of 0.056%, with USD→CAD→CNY→JPY→USD (shown in red) being the optimal arbitrage opportunity providing a profit of 0.074% [1.31904 * 5.10327 * 15.47 * 0.00961].



[Fig 2: The optimal arbitrage opportunity in G]

In terms of our graph, a feasible arbitrage opportunity is represented by a cycle such that when the weights of the edges in the cycle are multiplied, the result α is greater than one. The optimal arbitrage opportunity is given by the cycle for which α is greatest.

$$\alpha = W_{n,1} \times \prod_{i=1}^{n-1} W_{i,i+1} \quad > 1$$

Detecting the existence of arbitrage opportunities

A brute force method to detect arbitrage opportunities may involve using an algorithm to find all cycles in the graph, for example depth-first search. Each cycle is then tested for an arbitrage opportunity by multiplying the edge weights together and checking if the result is greater than one. The algorithm could terminate after finding one arbitrage opportunity, or check every cycle to find the optimal arbitrage opportunity.

With n currencies, the complete graph contains $\binom{n}{k}(k-1)!$ cycles of length k, and $\sum_{k=2}^{n}\binom{n}{k}(k-1)!$ cycles in total. Checking if a cycle of length k provides an arbitrage opportunity requires k multiplications. Hence a brute force algorithm would require $\sum_{k=2}^{n} k\binom{n}{k}(k-1)! = \sum_{k=2}^{n}\binom{n}{k}k! = \sum_{k=2}^{n}\frac{n!}{(n-k)!}$ operations, which is of factorial order $O(n!)$.

The process of detecting an arbitrage opportunity can be simplified by using the logarithm of the exchange rates as the weights on the graph's edges, instead of using the exchange rate itself. Due to the property $\lg(A) + \lg(B) = \lg(AB)$, arbitrage opportunities can be detected by adding the weights of the edges in the cycles as opposed to multiplying them.

An arbitrage opportunity is occurs where $W_{1,2}W_{2,3}\cdots W_{n-1,n}W_{n,1} > 1$, so with transforming the weights to $\lg(W_{ij})$ this condition becomes

$\lg(W_{1,2}W_{2,3}\cdots W_{n-1,n}W_{n,1}) > \lg(1) \implies \lg(W_{1,2}) + \lg(W_{2,3}) + \cdots + \lg(W_{n-1,n}) + \lg(W_{n,1}) > 0$, meaning that a positive weight cycle in the graph represents an arbitrage opportunity. The optimal arbitrage opportunity is given by the maximal positive weight cycle.

Furthermore, transforming the weights to $A_{ij} = -\lg(W_{ij})$, the condition becomes

$$-\lg(W_{1,2}W_{2,3}\cdots W_{n-1,n}W_{n,1}) < -\lg(1) \implies A_{1,2} + B_{2,3} + \cdots + A_{n-1,n} + A_{n,1} < 0\ ,$$

hence a negative weight cycle in the graph represents an arbitrage opportunity. The optimal arbitrage opportunity is given by the minimal negative weight cycle.

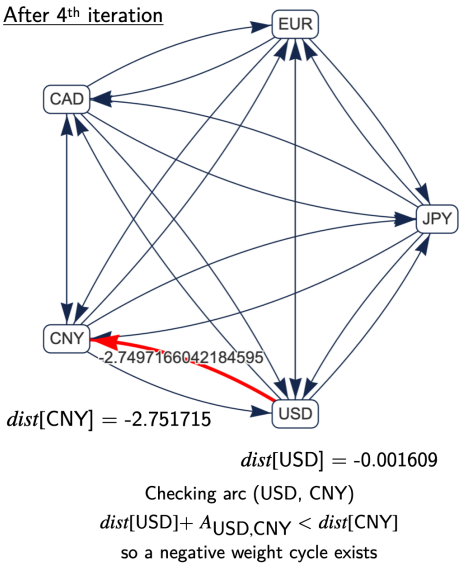
Having an additive, rather than multiplicative, condition to check for arbitrage allows us to apply standard graph algorithms to this problem, which often calculate distances by adding edge weights, not multiplying them. However, since $\lg(W_{ij})$ is negative for $0 < W_{ij} < 1$, algorithms that assume edge weights are positive, such as Dijkstra's algorithm, can not be used. Instead, algorithms such as Floyd-Warshall or Bellman-Ford can be applied, which have a worse time complexity but provide correct results with negative weights.

Using the Bellman-Ford algorithm to detect arbitrage opportunities

The Bellman-Ford algorithm provides the shortest distances from one node to any other node in a graph, and can be used to detect negative-weight cycles in the graph. This allows for the detection of arbitrage opportunities on the graph G', which is uses the negative log weights A_{ij}.

Iteration		USD	JPY	EUR	CAD	CNY
Initial	$dist$	0.000000	Inf	Inf	Inf	Inf
	$pred$	USD	None	None	None	None
1	$dist$	0.000000	-4.644872	0.000000	-0.276904	-1.905958
	$pred$	USD	USD	USD	USD	USD
2	$dist$	-0.000558	-4.645688	0.096292	-0.277462	-1.906786
	$pred$	CNY	CAD	USD	CAD	CNY
3	$dist$	-0.001115	-4.646246	0.095734	-0.278020	-1.907343
	$pred$	CNY	CNY	CAD	USD	CAD
4	$dist$	-0.001673	-4.646804	0.095177	-0.278577	-1.907901
	$pred$	CNY	CNY	CAD	USD	CAD

[Fig 3a: Iterations of the Bellman-Ford algorithm on G']



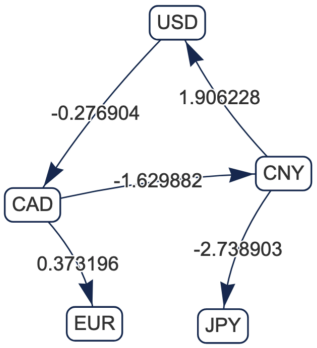
[Fig 3b: Detecting negative weight cycle]

The list $dist$ stores the shortest distance from the start node to every other node, initialised with zero for the start node (USD) and infinity for every other node. At each iteration, every edge (c_i, c_j) with weight A_{ij} in the graph is checked to see if the path to c_j using this edge is shorter than the currently identified shortest path to c_j. That is, $dist[j] \leftarrow dist[i] + A_{ij}$ if this value is smaller than the current $dist[j]$. The $pred$ list is also updated keep track of this edge (c_i, c_j) by storing the predecessor vertex i at $pred[j]$. Fig 4a shows this process on G'. [5].

In a graph without negative-weight cycles, this algorithm will run for at most $n - 1$ iterations since the longest path from the start node to any other node which could provide the shortest distance, of length $n - 1$, will have been checked by the algorithm and there will be no more opportunities to reduce the values in the $dist$ list.

However, if a negative weight cycle exists in the graph, it will be possible to continue reducing the values in the $dist$ list by taking a path that continually loops in this cycle, generating smaller and smaller shortest distances to these nodes. Hence, the existence of a negative-weight cycle can be determined by checking if a value in $dist$ could be reduced after the $(n-1)^{\text{th}}$ iteration, as seen in Fig 3b. To obtain the negative-weight cycle, the $pred$ list can be traversed in linear time to find the cycle formed between the vertices stored [6].

In our example, the $pred$ list defines a graph shown in Fig 3c.
The cycle detected from this list is USD→CAD→CNY→USD, with weight -0.000805. Reversing the negative log transformation, this corresponds to a profit of 0.056%, which is a valid arbitrage opportunity but not the optimal one.



[Fig 3c: Graph formed by $pred$ after the 4th iteration]

This method allows us to detect an arbitrage opportunity in polynomial time. The Bellman-Ford algorithm has a time complexity of $O(|V||E|)$ with $|V| = n$, and since the graphs for this problem are complete or near-complete, $|E| \sim O(n^2)$. Hence the algorithm has a time complexity of $O(n^3)$ for n currencies.

Finding optimal arbitrage opportunities

The Bellman-Ford algorithm will terminate on only one negative weight cycle in the graph, which may not provide the optimal arbitrage opportunity, as seen in the previous example. Finding an optimal cross-currency arbitrage opportunity is an NP-hard problem, since finding a maximum (or minimal) weight k-cycle in G' contains the travelling salesman problem as a special case (consider $k = |V|$) [11]. Binary integer linear programming can be used to solve this problem, which can be formulated as a maximum network flow optimisation problem [7].

Let x_{ij} be a binary variable to represent if edge (c_i, c_j) is included in the path (meaning c_1 is converted to c_2).

Maximise $\sum_{(i,j)\in E} x_{ij} \lg(W_{ij})$	[Maximise the log-conversion rate weight of currencies in the path]
Subject to $\sum_{j\neq i} x_{ij} = \sum_{j\neq i} x_{ji}$ for all $c_i \in V$	[Conservation of flow: At each node i, the number of edges included in the path incident to the node is equal to the number exiting the node. No transaction ends at node i, it is always converted to another currency, forcing a cycle]
$\sum_{j,(i,j)\in E} x_{ij} \leq 1$ for all $c_i \in V$	[At each node i, at most one edge included in the path can be incident on this node. This forbids passing through a node twice] [9]
$x_{ij} = 0$ or 1, for all $c_i, c_j \in V$, $i \neq j$	[x_{ij} is restricted to be a binary integer]

The constraints ensure that the edges included in the solution will form a cycle, which can be obtained by determining which variables x_{ij} are set to 1. An arbitrage opportunity exists if and only if the optimal objective value is positive.

To solve this problem, Soon and Ye [7] used the network simplex method, a variant of the simplex method designed to solve network flow problems. Furthermore, they used sensitivity analysis to determine in whether, upon receiving updated interest rate data, the current solution is still optimal or if further iterations of the network simplex method are required to re-optimise the solution. This allows the current solution to be updated quickly to detect new arbitrage opportunities as the exchange rates change in real time.

Quantum Computing in finance

Quantum computing exploits quantum mechanical properties of matter to perform calculations as opposed to the familiar bits, transistors, and logic gates of classical computing. This has the potential to disrupt fields such as logistics, cryptography, and finance due to the vast computational speedups quantum algorithms can provide for certain problems compared to their fastest classical counterparts.

Orús, Mugel and Lizaso [8] provided an overview into the state of quantum computing in finance, including the applications of quantum optimisation, quantum machine learning methods and quantum amplitude estimation (for Monte Carlo sampling) in finance. Some of their overview of quantum computing is provided before exploring its applications in finding optimal arbitrage opportunities.

In quantum computing, a qubit is the minimum amount of processable information, which encodes classical bits 0 and 1 as states $|0\rangle$ and $|1\rangle$. It is possible for the system to be in a superposition of these states, where the system is in all states simultaneously. Multiple qubits can be entangled, meaning that it is not possible to describe their state individually. Entanglement is exploited by quantum algorithms to obtain computational speedups over their classical counterparts, since systems without much entanglement can be efficiently modelled with classical methods such as tensor networks. Fig 4 shows the general structure of a quantum algorithm.

1. Encode the input data into the state of a set of qubits.
2. Bring the qubits into superposition over many states (i.e., use quantum superposition).
3. Apply an algorithm (or oracle) simultaneously to all the states (i.e., use quantum entanglement amongst the qubits); at the end of this step, one of these states holds the correct answer.
4. Amplify the probability of measuring the correct state (i.e., use quantum interference).
5. Measure one or more qubits.

[Fig 4: General structure of a quantum algorithm. Source: Orús Mugel and Lizaso [8]]

Quantum computers currently operate at relatively small scales, facing challenges such as decoherence (uncontrolled interactions between the system and its environment) and limitations in quantum error correction (which tries to correct for decoherence). While this makes it challenging for quantum computers to outperform classical computers in practice, there is a significant push towards advancement in the field with various technology and banking companies actively investing in the development of the technology.

Quantum Computing to find optimal arbitrage opportunities

Adiabatic quantum computing is a quantum computing model based on the idea that a quantum system which starts in a ground state will likely remain in a ground state throughout a slow deformation of the initial state. [10] This idea can be formalised using the Hamiltonian operator on quantum systems (which describes the total energy of the system at a given state), and proven to obtain the adiabatic quantum theorem.

Adiabatic quantum computing can be used to solve optimisation problems: the optimisation problem is mapped to the physical problem of finding the ground state of the Hamiltonian H_p, which is encoded with the objective function to minimise. The system is initialised with a known and easy-to-prepare ground state H_0. Over time, this system is deformed to H_p, which according to the quantum adiabatic theorem, finds with a high probability the ground state of H_p, providing the solution to the optimisation problem. Quantum annealers use adiabatic evolution to solve optimisation problems, with dedicated hardware being currently commercially available [8].

A similar classical algorithm is simulated annealing. To apply this to finding currency arbitrage opportunities, the algorithm starts with an initial cycle, and modifies the current solution by adding and/or removing edges to generate a neighbouring cycle. This cycle is evaluated against the objective function, and compared to the objective value of the current cycle. A temperature variable defines how likely the algorithm is to set the current solution to a worse neighbouring solution. At the start of the program, the temperature is high and it is more likely to move to a worse neighbouring solution, promoting *exploration*. As the iterations go on, the temperature is reduced, and the algorithm is less likely to move to a worse neighbouring solution, promoting *exploitation* of the current solution until the algorithm terminates.

Both simulated annealing and quantum annealing are examples of metaheuristic methods. Metaheuristic methods explore the optimisation function $f(x)$ by evaluating it at values various values of x, which are usually determined using the notion that good solutions are likely to be near other good solutions. Whilst simulated annealing explores the optimisation surface by 'walking over peaks', quantum annealing does this by 'tunnelling through peaks' [10]. In both cases, it is not guaranteed that the algorithm will find the optimal solution.

Rosenberg [9] reformulated the currency arbitrage as a Quadratic Unbounded Binary Optimisation (QUBO) problem to allow the D-Wave 2X quantum annealer to solve the problem, providing the solution in tens of milliseconds. This reformulation involved rewriting the constraints from the linear programming formulation as penalty terms on the objective function. The author also considered extra penalty terms to reduce the execution risk of the transaction (by incentivising shorter cycles) or terms to include fixed transaction costs. With the example used in this deep dive, which is taken from this paper, the annealer successfully found the optimal solution USD→CAD→CNY→JPY→USD which provides a profit of 0.074%.

Conclusion

Various techniques from depth-first search to quantum optimisation can be used to detect currency arbitrage opportunities. These methods have applied beyond foreign exchange markets, for example, negative cycle detection has been applied to detect arbitrage in decentralised cryptocurrency exchanges [11, recommended reading]. Further developments in this field aims to find better arbitrage opportunities more efficiently, with methods such as one involving operating on the graph's adjacency matrix [4] or a quantum-inspired simulated bifurcation approach [12] offering the reader further avenues for exploration.

As highlighted in the introduction, arbitrage is not risk-free in practice and the feasibility of identifying and profiting from detected opportunities may be impacted by market frictions and access to market data, which may be ignored by some of the theoretical methods. For example, Cai and Deng [13] have shown that the problem of merely detecting the existence of arbitrage in a frictional foreign exchange market is NP-complete (using a polynomial reduction from $\mathrm{SETCOVER}$), contrasting with our earlier polynomial-time solution with the Bellman-Ford algorithm. This underscores the importance of considering real-world complexities and market conditions when applying theoretical models to actual trading scenarios.

References

[1] Andrei Shleifer and Robert W. Vishny. "The Limits of Arbitrage." The Journal of Finance (1997)

[2] Robert W. Kolb, James A. Overdahl. "Futures, Options, and Swaps, 5th Edition" Chapter 1 Answers

[3] Akram, Qaisar Farooq and Rime, Dagfinn and Sarno, Lucio, "Arbitrage in the Foreign Exchange Market: Turning on the Microscope." Norges Bank Working Paper No. 2005/12, EFA 2006 Zurich Meetings

[4] Zhenyu Cui and Stephen Taylor. "Circular Arbitrage Detection Using Graphs." Stevens Institute of Technology School of Business Research Paper (2018)

[5] Robert Sedgewick and Kevin Wayne "Algorithms, 4th edition" 4.4 Shortest Paths

[6] Nina Amenta. Solutions HW5 ECS 122A. https://www.cs.ucdavis.edu/~amenta/f05/hw5.pdf

[7] Wanmei Soon and Heng-Qing Ye "Currency Arbitrage Detection Using A Binary Integer Programming Model." 2007 IEEE International Conference on Industrial Engineering and Engineering Management

[8] Orus, Roman, Samuel Mugel, and Enrique Lizaso. "Quantum computing for finance: Overview and prospects." Reviews in Physics 4 (2019)

[9] Rosenberg, G. "Finding optimal arbitrage opportunities using a quantum annealer." 1QB Information Technologies Write Paper (2016)

[10] Steven Herbert. "Lecture 15: Adiabatic Quantum Computing". Quantum Computing (CST Part II) (2019)

[11] Samuel Grone, Weitian Tong, Hayden Wimmer and Yao Xu. "Arbitrage Behavior amongst Multiple Cryptocurrency Exchange Markets." 2021 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA (2021).

[12] Kosuke Tatsumura, Ryo Hidaka, Masaya Yamasaki, Yoshisato Sakai, and Hayato Goto. "A Currency Arbitrage Machine based on the Simulated Bifurcation Algorithm for Ultrafast Detection of Optimal Opportunity." IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain (2020)

[13] Mao-cheng Cai, Xiaotie Deng. "Approximation and Computation of Arbitrage in Frictional Foreign Exchange Market (Extended Abstract)." Electronic Notes in Theoretical Computer Science (2003)