Arbitrage Detection Using the Bellman-Ford Algorithm

Jacob Dell

December 2025

1 Introduction

Arbitrage is the practice of taking advantage of price differences between markets. In currency exchange, an **arbitrage cycle** occurs when you can start with one currency, exchange through a series of other currencies, and end up with more of the original currency than you started with. Detecting such opportunities can be modeled as a graph problem.

The **Bellman-Ford algorithm** is a well-known method for finding the shortest paths from a single source in a weighted graph, even when some edges have negative weights. It has a runtime of $O(VE)$, where V is the number of vertices and E is the number of edges. Unlike Dijkstra's algorithm, Bellman-Ford can handle negative weights and, importantly, can detect negative-weight cycles—a feature that makes it especially useful for spotting arbitrage opportunities.

2 Bellman-Ford Algorithm

The algorithm works as follows:

1. Initialize the distance to all vertices as infinity, except the source vertex which is set to zero.
2. Relax all edges $V-1$ times, where V is the number of vertices. Relaxing an edge (u,v) means updating:
$$\text{distance}[v] = \min(\text{distance}[v], \text{distance}[u] + w(u,v))$$
where $w(u,v)$ is the weight of the edge from u to v.
3. After $V-1$ iterations, perform one additional iteration. If any distance can still be relaxed, a negative-weight cycle exists in the graph.

Time Complexity: Each iteration relaxes all E edges, and there are $V-1$ iterations plus one extra iteration for cycle detection. Therefore, the complexity is $O(V \cdot E)$.

3 Arbitrage as a Graph Problem

3.1 Example of Currency Arbitrage

Suppose we have three currencies: USD, EUR, and GBP, with the following exchange rates:

- USD to EUR: 0.9
- EUR to GBP: 0.8
- GBP to USD: 1.5

Starting with 1 USD:

$$1 \text{ USD} \to 0.9 \text{ EUR} \to 0.9 \cdot 0.8 = 0.72 \text{ GBP} \to 0.72 \cdot 1.5 = 1.08 \text{ USD}$$

We end up with more USD than we started with, creating an arbitrage opportunity.

3.2 Graph Construction

To detect arbitrage:

1. Represent each currency as a vertex.
2. Represent exchange rates as directed edges. Let the edge weight from currency i to j be:

$$w(i,j) = -\log(\text{exchange rate from } i \text{ to } j)$$

Why negative logarithm? If an arbitrage cycle exists:

$$r_1 \cdot r_2 \cdot \dots \cdot r_k > 1$$

Taking the logarithm:

$$\log(r_1) + \log(r_2) + \dots + \log(r_k) > 0$$

Multiplying by -1 converts this to:

$$-\log(r_1) - \log(r_2) - \dots - \log(r_k) < 0$$

So detecting arbitrage reduces to finding a **negative-weight cycle** in the graph.

3.3 Using Bellman-Ford for Arbitrage Detection

1. Build the graph with vertices as currencies and edges as $-\log(\text{exchange rate})$.
2. Pick an arbitrary source vertex and run Bellman-Ford.
3. If Bellman-Ford detects a negative cycle, an arbitrage opportunity exists.

4 Example Graph Construction

Suppose we have two currencies, A and B, with exchange rates:

- A to B: x/y
- B to A: y/x

Construct the graph:

- Vertex set: $\{A, B\}$
- Edge weights: $w(A, B) = -\log(x/y)$, $w(B, A) = -\log(y/x)$

A cycle $A \to B \to A$ forms a negative cycle if:

$$w(A, B) + w(B, A) = -\log(x/y) - \log(y/x) = -\log(1) = 0$$

If there are more currencies and the product of rates along a cycle exceeds 1, the sum of negative logs becomes negative, indicating an arbitrage opportunity.

5 Conclusion

Arbitrage detection in currency exchange can be effectively solved using graph algorithms. By transforming exchange rates into negative logarithms and applying the Bellman-Ford algorithm, one can efficiently detect profitable cycles, even in the presence of complex currency networks.