“Triangular” arbitrage of cryptocurrencies

Kumaran Nathan
www.linkedin.com/in/kumarannathan

Algorithmic Trading, London
24th April 2024

Algorithmic trading amateur

- This was my **first time developing a trading strategy**
  - Only ever been a ‘buy and hold’ investor in equity index funds
- This was my **first time creating an automated trading system**
  - No prior experience with exchanges / APIs
  - No prior experience with cloud computing platforms
- This was my **first time trading cryptocurrencies**

- Programming experience from engineering undergrad and PhD
- Worked as an electrical engineer, strategy consultant, and telco director
- Never worked in any kind of quant role / hedge fund / HFT or MM firm

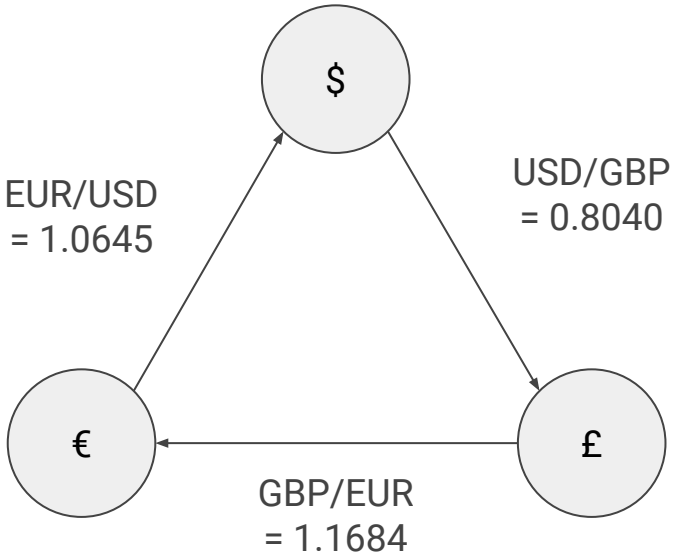
Exchange rates as of 18th April 2024

Common types of arbitrage in cryptocurrencies

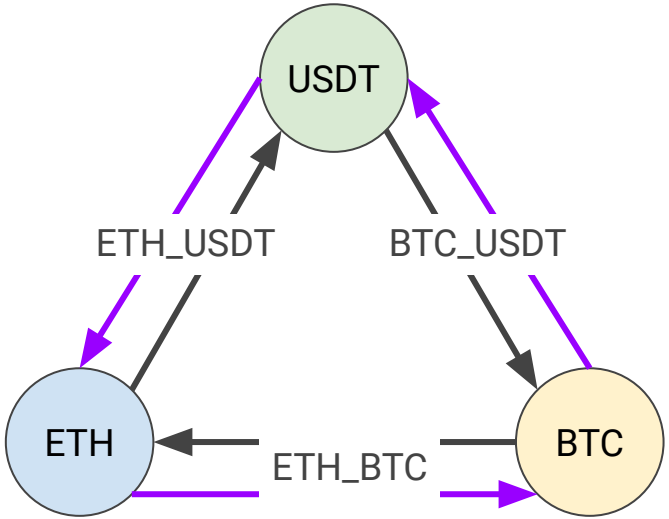
Cross-exchange arbitrage:

- Price difference of same asset on different exchanges
- E.g., Sam Bankman-Fried bought BTC for $10k in the US and sold this for $11.5k worth of Japanese yen

Triangular arbitrage:

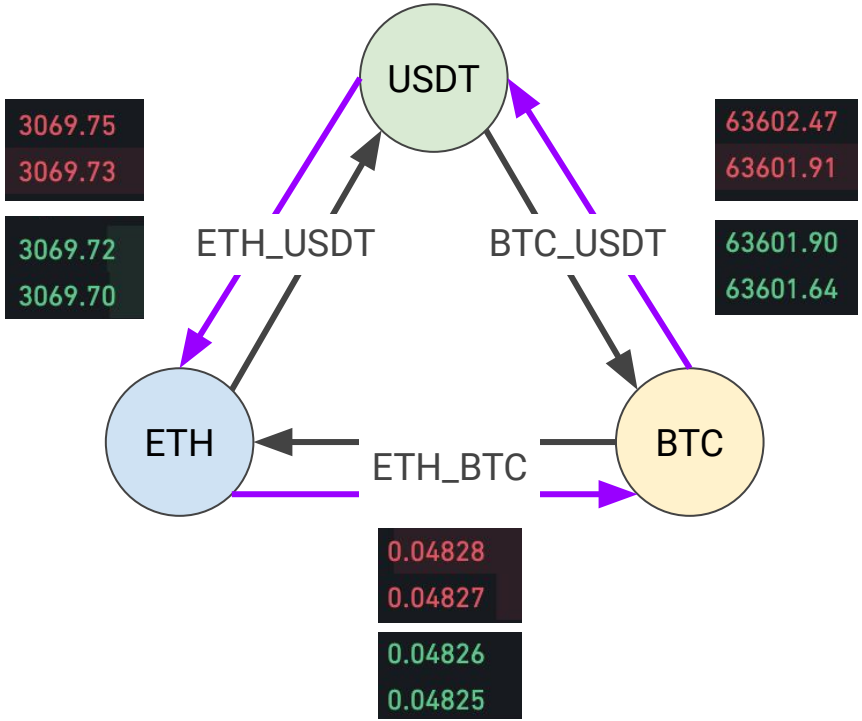


Triangular arbitrage applied to cryptocurrencies



- **Bidirectional**
  - Clockwise with black arrows
  - Anti-clockwise with purple arrows
- **Liquidity taker**
  - Buy @ ask price, plus trading fees
  - Sell @ bid price, minus trading fees
- **Theoretically risk-free**, though in reality:
  - Execution risk: order book has moved before orders have reached matching engine
  - Counterparty risk
- No separate “exit” (unlike most other strategies)
- Did this during Binance’s ‘zero spot trading fees’ promotion

Real example



- **Spreads are only 1 tick and fees are zero**
- Clockwise
  - USDT → BTC
    - Buy BTC_USDT @ ask
    - 1/63601.91
  - BTC → ETH
    - Buy ETH_BTC @ ask
    - 1/0.04827
  - ETH → USDT:
    - Sell ETH_USDT @ bid
    - 3069.72
  - Multiplying these gives **0.9998878864812**
- Anticlockwise
  - Same approach gives **0.99990152032915**
- **Volume limited by smallest edge**

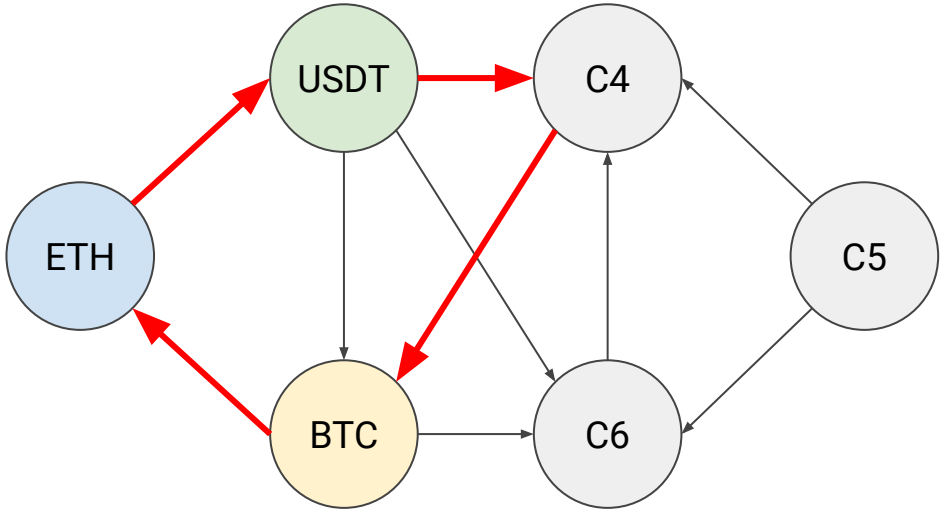
Creating an automated trading system

- MVP system had **WebSocket streams for price & quantity data** and **order placement via REST APIs**
- Post-trade analysis logged to database outside of hot path (order context, orders, fills, PnL) and analysed with PowerBI dashboards
- This approach identified and reacted to opportunities however **many orders were filled as maker** (not taker) indicating that **markets had moved by the time my orders reached the matching engine**

The need for speed (or reduced latency):

- Moved system to EC2 instance hosted in same AWS region as Binance's matching engine (Tokyo) & availability zone
- Placing sequential orders is slow so I wanted to place the orders concurrently. This meant I needed to maintain a balance of each asset (and therefore had exposure to underlying price)
- Only streamed tops of order books rather than maintaining full L2 order book (few inefficiencies at depth)
- Compiled computationally intensive sections of code to C using Cython
- Use breakeven limit prices to improve fill rates

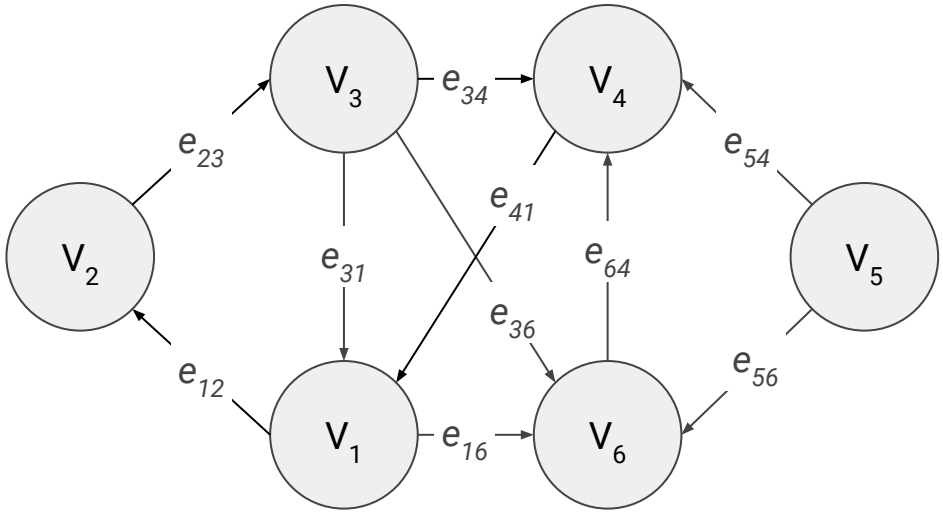
Expanding beyond the “USDT-BTC-ETH” triangle



Only showing 10 edges here for simplicity, but in practice there would be 30 edges if all currencies are tradable pairs

- Wanted to **cover more triangles**, and **cycles with a higher number of edges**
- This example of **6 currencies has 394 paths**:
  - 3 edges: 40 paths
  - 4 edges: 90 paths
  - 5 edges: 144 paths
  - 6 edges: 120 paths
- With **10 currencies this rises to over a million possible paths**
- **More edges in loops = higher spreads and trading fees** so less likely to find a profitable path

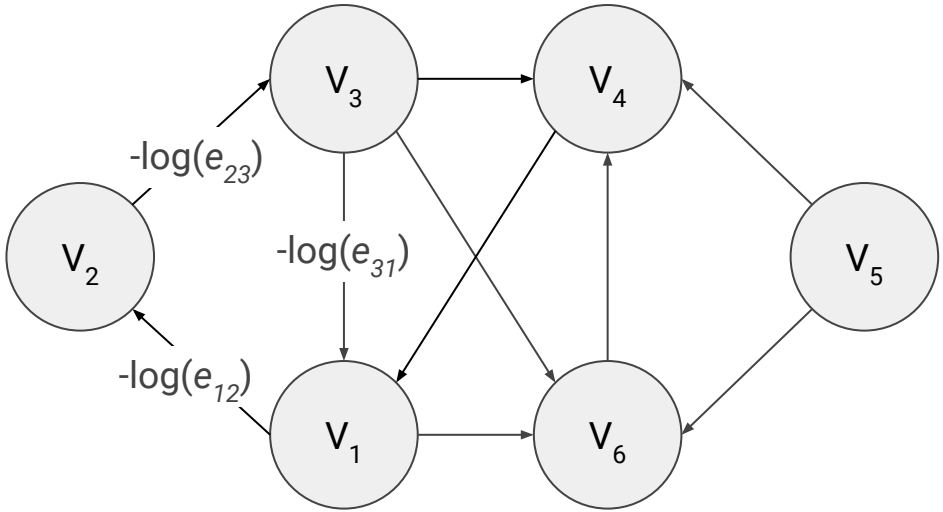
Modelling generalised scenarios using graph theory



- Use a **directed graph** as edges are weighted differently in each direction due to bid-ask spread
  - Currencies become vertices (V_1, V_2, .. V_n)
  - The edge e_{ij} runs from V_i to V_j and represents the exchange rate between these currencies
- Arbitrage opportunity of $V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_1$ if:

$$e_{12} \times e_{23} \times e_{31} > 1$$

Solving this as a ‘shortest path’ problem



- Finding shortest paths is a common problem in graph theory and computer science, minimising the sum of the edge weights between nodes along a path

  $\mathbf{e_{12} \times e_{23} \times e_{31} > 1}$

  $\log(e_{12} \times e_{23} \times e_{31}) > \log(1)$

  $\log(e_{12}) + \log(e_{23}) + \log(e_{31}) > 0$

  $\mathbf{-\log(e_{12}) + -\log(e_{23}) + -\log(e_{31}) < 0}$

- Identifying arbitrage opportunities is equivalent to **finding negative weight cycles**
- Examples of algorithms to find the shortest path in a weighted graph include Dijkstra's algorithm, **Bellman-Ford**, and **Shortest Path Faster Algorithm**

And we’re only just scratching the surface…

- **Managing asset balances** over time (the minimum balance is the limiting factor for concurrent trades)
  - Changes due to asset price changes
  - Changes due to arbitrage
- **Handling unfilled orders**
  - Cancel? After how long?
  - Fill at market price?
  - Something else?
- Using **different order types** (limit vs. market)
- **Using multiple exchanges** (though challenge and costs associated with managing balances)

Best results from 2022

- The **theoretical PnL graph should be monotonically non-decreasing** but in practice it can be like **“picking up pennies in front of a steamroller”**
- **Not scalable due to low capacity** so deployed minimal capital (each dollar traded >100k times)
- My peak month had a **trading volume of well over $100MM**
- Profit **margin is a tiny fraction of a percent** (otherwise I’d be very rich from this!)
- **Strategy’s viability killed overnight** with the end of zero-fee trading promotion
- Still possible to monitor opportunities now and can check viability based on fee rate
    - Managed to **mock this up in ProfitView in a few minutes and with less than 60 lines of code**, monitoring three exchanges
    - Still detects some opportunities but these are relatively rare