Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 3, June 2026, pp. 2579~2594
ISSN: 2302-9285, DOI: 10.11591/eei.v15i3.10817

An innovative approach to identifying triangular arbitrage opportunities in financial markets using the Bellman-Ford algorithm

Issam Akouaouch[1,3], Anas Bouayad[2,3]
[1]Applied Physics, Computer Science and Statistics Laboratory, Sidi Mohamed Ben Abdellah University, Fez, Morocco
[2]Artificial Intelligence, Data Sciences and Emerging Systems Laboratory, Sidi Mohamed Ben Abdellah University, Fez, Morocco
[3]Department of Computer Science, Faculty of Science, Sidi Mohamed Ben Abdellah University, Fez, Morocco

Article Info

Article history:

Received Jun 8, 2025
Revised Mar 15, 2026
Accepted Apr 1, 2026

Keywords:

Algorithmic trading
Bellman-Ford algorithm
Financial markets
High-frequency trading
Triangular arbitrage

ABSTRACT

Existing arbitrage detection techniques rely on exhaustive search or linear programming, which are computationally expensive and often miss profitable cycles in dynamic markets. Triangular arbitrage is a profitable trading strategy that exploits discrepancies in currency exchange rates, but common algorithms detect only a limited number of loops and cannot find non-loop opportunities. To address these gaps, this study presents a real-time, graph-based framework for identifying triangular arbitrage opportunities in cryptocurrency markets using an optimized implementation of the Bellman–Ford algorithm. By modeling currency exchange rates as a directed graph and detecting negative-weight cycles, the framework efficiently identifies profitable arbitrage opportunities under realistic trading conditions. The proposed framework achieves an average detection latency of 0.002 milliseconds, providing empirical performance benchmarks for single-exchange cryptocurrency trading systems. Experiments on a six-month historical dataset yielded a detection accuracy of 92%, while additional validation on live cryptocurrency market data streams confirmed the framework's real-time performance and low latency. This high-speed detection is crucial in high-frequency trading (HFT), where brief pricing inefficiencies can yield significant profits before being corrected. The experimental pipeline is designed to support reproducibility and comparative evaluation in applied FinTech research.

Corresponding Author:

Issam Akouaouch
Applied Physics, Computer Science and Statistics Laboratory, Sidi Mohamed Ben Abdellah University
Fez, Morocco
Email: Issam.akouaouch@usmba.ac.ma

1. INTRODUCTION

Triangular arbitrage is a trading strategy that leverages price discrepancies between three or more currencies to secure risk-free profit. By sequentially converting an initial currency into a second, then a third and finally back into the original currency, traders can exploit temporary misalignments in exchange rates [1]. When exchange rates fail to reflect accurate value relationships, these cycles allow investors to obtain risk-free profits during brief windows of market inefficiency [2]. Such inefficiencies are typically short-lived in highly competitive financial markets, which means detection methods must be both precise and fast [3].

Over the past decade, researchers have proposed various models to identify triangular arbitrage opportunities. Early algorithms relied on exhaustive search and linear programming techniques, which struggle to keep up with the computational demands of real-time trading [4]. The widely used Moore–Bellman–Ford (MBF) algorithm can detect only a limited number of arbitrage loops per run and cannot identify non-loop arbitrage paths [5]. Additionally, several studies focused on measuring arbitrage profits

rather than improving detection speed and accuracy [6], leaving a gap in the literature for methods that are both efficient and comprehensive.

Among graph-based algorithms, the Bellman–Ford algorithm is particularly suited to arbitrage detection because it identifies negative-weight cycles in directed graphs. Unlike Dijkstra's algorithm, which assumes non-negative edge weights, Bellman–Ford remains applicable when log-transformed exchange rates and transaction costs yield negative weights [7]. Its O(VE) computational complexity and compatibility with dynamic graph updates make it suitable for high-frequency trading (HFT) environments [8]. These characteristics motivate its selection over alternative shortest-path methods in latency-sensitive financial applications.

Unlike previous methods that either rely on specialized hardware (for example, graphics processing units (GPUs) or graph neural networks (GNNs)) or focus on detecting only non-loop paths, our approach explicitly integrates transaction costs and latency constraints into the Bellman–Ford framework. This integration yields a lightweight yet effective solution for real-time triangular arbitrage detection, improving both speed and accuracy without requiring training data or exotic hardware.

This paper presents an optimized real-time implementation that models currency exchange rates as a weighted directed graph and applies the Bellman–Ford algorithm to detect triangular arbitrage opportunities. The framework incorporates transaction costs, latency considerations, and market volatility into the detection process to ensure that only executable and profitable cycles are identified. The contributions of this work are fourfold. First, we provide a structured review of existing arbitrage detection methods and their practical limitations. Second, we present an applied graph-based framework for real-time triangular arbitrage detection. Third, we conduct an empirical evaluation against benchmark methods using real exchange data. Finally, we analyze practical deployment considerations.

The contribution of this work is primarily practical rather than theoretical. We do not introduce a new shortest-path algorithm, nor do we modify the theoretical properties of the Bellman–Ford algorithm. Instead, this study demonstrates how a carefully engineered implementation, combined with realistic modeling of transaction costs, latency, and streaming data constraints, can achieve reliable performance in real-time arbitrage detection. The emphasis of this work is on system design, empirical benchmarking, and execution-oriented evaluation. The proposed framework is evaluated using both historical market data for reproducible backtesting and live data streams to validate real-time operation under realistic trading conditions.

The remainder of this paper is organized as follows. Section 2 reviews related work and details the proposed methodology. Section 3 describes our data collection and pre-processing procedures. Section 4 presents the experimental setup, including evaluation metrics and baseline methods. Section 5 reports the results and analysis, while section 6 discusses their broader implications, limitations, and potential extensions. Section 7 concludes by summarizing our contributions and suggesting directions for future research.

2. METHOD

2.1. Related work

The identification of triangular arbitrage opportunities has evolved significantly with advancements in computational finance and algorithmic trading. Early foundational work by Fenn *et al.* [9] rigorously analyzed triangular arbitrage in foreign exchange (Forex) markets, establishing theoretical frameworks for detecting risk-free profit cycles. However, their model assumed static exchange rates and neglected transaction costs, limiting its practical utility in dynamic markets.

The Bellman-Ford algorithm, first formalized in Cormen *et al*. [10], has been widely recognized for its ability to detect negative cycles in graphs, a critical requirement for arbitrage detection. While Cormen *et al*. [10] laid the theoretical groundwork, Harish and Narayanan [11] later demonstrated its computational advantages over Dijkstra's algorithm in sparse graphs, particularly for scenarios involving frequent exchange rate updates. This adaptability made Bellman-Ford a preferred choice for modeling real-time financial networks.

With the rise of HFT, researchers began focusing on latency reduction and scalability. Dixon *et al*. [12] explored GPU-accelerated graph processing to parallelize pathfinding algorithms, significantly improving processing speeds for large currency networks. However, their reliance on specialized hardware limited accessibility for mainstream trading platforms. Concurrently, Nevmyvaka *et al*. [13] proposed reinforcement learning techniques to optimize trade execution. While not directly focused on arbitrage detection, their work demonstrates how predictive models can improve execution precision in dynamic environments.

Garleanu and Pedersen [14] identified a key shortcoming in arbitrage literature: the omission of transaction costs, which can significantly distort expected profits if left un-accounted. Likewise, Aldridge

[15] emphasized the necessity for adaptive trading algorithms that respond dynamically to market liquidity changes and evolving regulatory environments.

Recent research has explored machine learning and enhanced graph techniques to overcome the limitations of traditional arbitrage detection. Zhang [4] introduced a GNN framework for triangular arbitrage detection that models the currency exchange network as a graph and uses deep learning to efficiently identify profitable arbitrage opportunities. In experiments, their GNN-based method achieved an average yield of 6.3% with a computational time of 147 ms per network, compared to 5.8% yield and 215 ms for the Bellman–Ford algorithm and 6.0% yield and 320 ms for a linear-programming solver [4]. However, the approach requires training data and specialized hardware. In parallel, Zhang *et al*. [5] proposed a modified Moore–Bellman–Ford (MMBF) algorithm online graphs to detect both arbitrage loops and non-loop paths on decentralized exchanges. They showed that the MMBF algorithm uncovers more arbitrage opportunities than the standard MBF algorithm, although the detected paths may not always maximize profitability and gas fees were not incorporated.

This paper addresses these gaps by optimizing the Bellman-Ford algorithm to handle dynamic transaction costs and latency-sensitive environments, offering a robust and practical solution for modern trading systems, Table 1 summarizes the key contributions and limitations of relevant prior studies in arbitrage detection and market analysis.

Table 1. Summary table of key contributions (additional studies)

Study	Methodology	Key contribution	Limitation
Easley *et al.* (2011) [16]	Market microstructure analysis	Identified arbitrage opportunities during the flash crash using order book data	Limited to single-exchange scenarios
Menkveld (2013) [17]	Empirical analysis of HFT behavior	Demonstrated how high-frequency traders function as liquidity providers in modern markets	Did not model behavior via agent-based simulation
Avellaneda and Stoikov (2008) [18]	Stochastic control modeling	Developed optimal market-making strategies under inventory and volatility constraints	Does not use reinforcement learning
Almgren and Chriss (2001) [19]	Dynamic programming	Integrated transaction costs into execution strategies for portfolio optimization	Assumes fixed transaction cost structures
Kirilenko *et al.* (2017) [20]	Flash crash market analysis	Investigated HFT impact on price dynamics during extreme events	Does not address distributed arbitrage or multi-exchange synchronization
Gatev *et al.* (2006) [21]	Statistical arbitrage (cointegration)	Developed a relative-value pairs trading strategy to detect temporary mispricings	Designed for long-term strategies, not HFT
Cui and Taylor (2018) [22]	Graph-theoretic analysis (graph cycles, circular arbitrage detection)	Introduced graph-theoretic methods for detecting circular arbitrage across multiple currencies and emphasized use of directed cycles to find profitable loops	Working paper, not limited to DEX but lacking empirical benchmarking in live crypto markets
Zhang (2025) [4]	GNN-based detection	Introduced a GNN approach for triangular arbitrage detection, achieving 6.3% yield and 147 ms versus 5.8% yield and 215 ms for Bellman–Ford	Requires training data and specialized hardware and does not focus on HFT
Zhang *et al.* (2024) [5]	Line-graph and MMBF algorithm	Developed a MMBF algorithm using line graphs to detect arbitrage loops and non-loop paths	Detected paths may not maximize profitability and gas fees were not considered

2.2. System model and problem formulation

To identify triangular arbitrage opportunities, we model the market as a directed weighted graph $G = (V, E)$. Each currency is a node $v \in V$. A directed edge $(u, v) \in E$ represents the act of converting currency u to currency v using the currently available executable price stream [23].

Exchange-rate representation: let $r_{uv}(t)$ denote the executable conversion rate from u to v at time t. When a venue streams bid/ask quotes, we map trades to quotes as follows: converting uv uses the bid of the pair quoted in units of v per u, converting vu uses the corresponding ask [24]. When only mid-prices are available, we apply the venue's published spread to obtain synthetic bid/ask bounds before execution checks.

Cost-adjusted edge weights: following standard practice, we log-transform rates so that multiplicative returns become additive along paths. For edge (u, v) at time t, the cost-adjusted weight is defined as,

$$w(u, v, t) = -\ln(r_{uv}(t) \cdot (1 - fee_{uv}) \cdot (1 - slip_{uv}(q, t))) \quad (1)$$

where fee_{uv} is the taker/maker fee factor and $slip_{uv}(q, t)$ captures price impact for notional size q based on available order-book depth. If depth is insufficient, the edge is temporarily ineligible.

Arbitrage condition: for a directed cycle $C = (v_0, v_1, v_2, v_0)$, define the cycle sum,

$$W(C,t) = \sum_{(x,y)\in C} w\,(x,y,t) \tag{2}$$

The classic no-arbitrage condition $r_{xy}(t) = \frac{1}{r_{yx(t)}}$, and more generally,

$$\prod_{(x,y)\in C} r_{xy}(t) = 1 \tag{3}$$

which implies $W(C,t) = 0$ for all cycles. Thus, a profitable triangular arbitrage (after fees and slippage) exists if and only if $W(C,t) < 0$. For a starting notional q in base currency v_0, the executed output and profit are:

$$q_{\text{out}}(C,t) = q\, e^{-W(C,t)} \tag{4}$$

$$\text{PnL}(C,t) = q_{\text{out}}(C,t) - q \tag{5}$$

Temporal validity: to mitigate stale-quote risk, we require a monotone timestamp sequence along a candidate cycle and enforce a maximum quote age $\Delta t_{\max}$. Quotes violating freshness are discarded before graph updates.

Liquidity and size constraints: we bound the trade notional q by the minimum executable depth across edges in C and reject cycles that cannot be filled atomically at the computed $slip_{uv}(q,t)$. This ensures that detected opportunities remain executable in practice [25].

Event-driven updates: the graph is maintained as an adjacency list keyed by source currency. On each tick, we update affected edges' weights $w(u,v,t)$ and, if requested in section 2.3, trigger the negative-cycle check starting from the touched nodes. This representation supports low-latency relaxations and aligns with the streaming data model used later.

Notation recap: Table 2 lists the symbols used in the graph model and adds the execution-specific quantities.

Table 2. Notation used in graph model

Symbol	Description
$\mathrm{G} = (V,E)$	Directed graph with currencies (V) and edges (E) representing exchange rates
$r_{uv}(t)$	Executable exchange rate (bid/ask) from currency u to v at time t
$w(u,v,t)$	Cost-adjusted log-weight for edge (u,v) at time t (see (1))
$d[v]$	Distance (shortest-path) estimate to node v from the source during Bellman–Ford relaxation
C	A directed cycle of currency exchanges (arbitrage loop)
fee_{uv}	Transaction-fee factor applied on trade $u \to v$
$slip_{uv}(q,t)$	Slippage factor for trade of size q at time t (price impact)
$\Delta t_{\max}$	Maximum allowed age of a quote for a valid cycle
q	Notional trade size in the base currency

2.3. Algorithmic implementation and complexity analysis

The Bellman–Ford algorithm is adapted to our financial context as follows. The currency exchange network is represented using an adjacency list, where each node stores its outgoing edges with log-transformed weights. Two arrays are maintained: $dist[]$ for tentative shortest distances and $pred[]$ for predecessor pointers. The algorithm proceeds in three stages:

a. Initialization—all distance estimates are set to $+\infty$, except for the base currency (source) which is set to 0. Predecessor pointers are initialized to a sentinel value $null$.
b. Relaxation—for $|V| - 1$ iterations, every edge (u,v) is examined. If,

$$dist[u] + w(u,v,t) < dist[v]$$

then $dist[v]$ and $pred[v]$ are updated. Using adjacency lists ensures $O(|V| \cdot |E|)$ time complexity, consistent with classical analyses [10].
c. Negative-cycle check and cycle extraction—a final scan over the edges tests for further improvement. If,

$$dist[u] + w(u,v,t) < dist[v]$$

a negative-weight cycle exists. The arbitrage loop is reconstructed by backtracking through $pred[]$ pointers until a node repeats. This procedure follows standard negative-cycle recovery techniques [26].

Algorithm: MBF for triangular arbitrage detection

a. Initialize: for each $v \in V$, set $dist[v] = +\infty$ and $pred[v] = null$ Choose a source s and set $dist[s] = 0$.
b. Relaxation: repeat $|V| - 1$ times: for each edge $(u, v) \in E$, compute cost-adjusted weight $w(u, v, t)$ (with fees and slippage, see (1)). If $dist[u] + w(u, v, t) < dist[v]$ update $dist[v]$ and set $pred[v] = u$.
c. Negative-cycle detection: for each $(u, v) \in E$, if $dist[u] + w(u, v, t) < dist[v]$, a negative cycle exists. Backtrack via $pred[]$ pointers to recover the cycle.
d. Output: return the cycle and compute net return using section 2.2 equations.

Practical considerations: two modifications are introduced for HFT. First, transaction fees and slippage are modeled explicitly in $w(u, v, t)$ (1), ensuring that only profitable cycles after costs are flagged [14], [19]. Second, the relaxation phase is bounded by the connected component size, with early termination if no updates occur in an iteration. This reduces average-case complexity in sparse graphs [11], [27].

The worst-case complexity remains $O(|V| \cdot |E|)$ [10] but early stopping and adjacency-list iteration significantly reduce runtime in practice. Bellman–Ford is chosen over faster algorithms such as Dijkstra's because it can detect negative-weight cycles. Dijkstra assumes non-negative weights and cannot identify arbitrage loops. Variants like improved MBF [5] and graph-neural-network approaches [4] show promise but require specialized hardware or cannot model dynamic fees. Our baseline implementation provides a transparent and reproducible reference for future research.

2.4. Application of the Bellman-Ford algorithm for triangular arbitrage detection

In practice, we integrate the modified Bellman–Ford algorithm into a real-time trading system. Whenever a new price update arrives, the corresponding edge weight $w(u, v, t)$ is updated and a cycle-detection routine is triggered starting from the affected nodes. The algorithm iteratively relaxes edges using cost-adjusted weights until no further improvement occurs or until all vertices ($|V| - 1$ iterations) have been traversed [10]. Negative-weight cycles are detected and reconstructed by following predecessor pointers. Related GNN approaches have also been proposed for triangular arbitrage detection [4], [28]. The expected output $q_{\text{out}}(C, t)$ and profit $\text{PnL}(C, t)$ are computed as defined in section 2.2. To mitigate latency, we maintain a queue of updated currencies so that only affected paths are revisited, and we enforce timestamp freshness and liquidity constraints during cycle extraction.

Figure 1 illustrates a typical triangular arbitrage cycle within our graph model. Pair currencies are represented as nodes (USDT, FIS, BTC), and directed edges correspond to executable exchange rates. The algorithm identifies the cycle USDT→FIS→BTC→USDT, which yields a net positive return once fees and slippage are taken into account. In practice, we record the detection latency and ensure that all three quotes are contemporaneous before executing the trade.

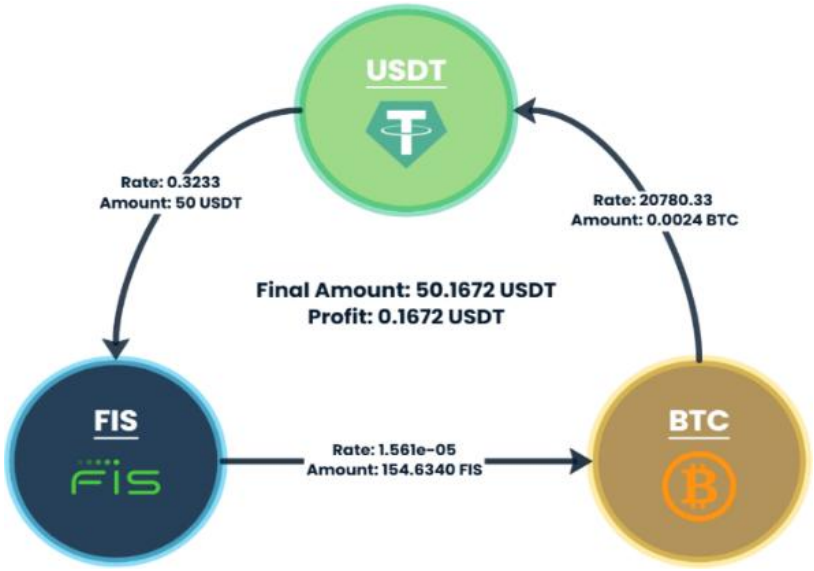


Figure 1. Graph representation of triangular arbitrage opportunity

2.5. Advantages of the proposed approach

The proposed approach offers several advantages for identifying triangular arbitrage opportunities. By utilizing the Bellman-Ford algorithm, we leverage its efficient computation, making it suitable for real-time analysis of currency exchange rates [29]. The algorithm's time complexity is $O(|V|\cdot|E|)$, where $|V|$ represents the number of nodes (currencies) in the graph and $|E|$ the number of edges (exchange rate pairs). This computational efficiency allows for swift detection of arbitrage opportunities, essential for success in fast-paced financial markets where timing is critical. Details on specific speed for identifying arbitrage opportunities and its implications are discussed in the results section.

Moreover, the graph representation of currency exchange rates provides a comprehensive view of the market, capturing the interdependencies between currencies [30], [31]. This enables us to detect arbitrage opportunities that may not be apparent in individual currency pairs but become evident when considering the entire network. The algorithm's ability to detect negative weight cycles ensures that potential triangular arbitrage opportunities are identified accurately [32].

Furthermore, the proposed approach can adapt to changing market conditions. As currency exchange rates fluctuate, the graph representation can be dynamically updated, allowing for real-time detection of new arbitrage opportunities or changes in existing ones. This adaptability enhances the effectiveness and relevance of the methodology in the highly competitive world of financial trading.

2.6. Considerations and limitations

While the proposed approach offers promising benefits, it is essential to consider certain limitations. One key consideration is the availability and reliability of real-time currency exchange data. Timely and accurate data are crucial for the success of the approach, as outdated or erroneous rates can lead to inaccurate identification of arbitrage opportunities [33]. Therefore, using reputable data sources and implementing data quality checks are essential to mitigate this limitation [34].

Another factor to consider is the assumption of frictionless markets, which may not hold. Transaction costs, liquidity constraints, and market regulations can impact the profitability and feasibility of executing triangular arbitrage strategies. Integrating such factors into the methodology may enhance its practical applicability.

Furthermore, the scalability of the approach should be considered, especially when dealing with large-scale currency markets. As the number of currencies and exchange rate pairs increases, the computational complexity of the Bellman-Ford algorithm may become a limiting factor. Exploring optimization techniques or alternative algorithms may be necessary to handle such scenarios [35].

3. DATA COLLECTION

3.1. Historical dataset for backtesting

For reproducible evaluation and controlled benchmarking, we constructed a historical dataset derived from cryptocurrency market data provided by the Binance exchange. The dataset consists of approximately six months of tick-level observations, including bid and ask price updates for a selected set of actively traded cryptocurrency pairs.

The historical data enables systematic backtesting of triangular arbitrage detection accuracy, execution latency, and profitability under a wide range of market conditions. By replaying recorded market states, the framework can be evaluated consistently across identical input sequences, allowing fair comparison with baseline methods and sensitivity analysis with respect to fees and latency assumptions.

All historical data are processed using the same graph construction and arbitrage detection pipeline employed in the live system, ensuring that backtesting results remain representative of real deployment conditions.

3.2. Real-time data stream evaluation

In addition to historical backtesting, the proposed framework was evaluated in a live trading environment using real-time cryptocurrency market data streams. The system connects directly to the Binance WebSocket application programming interface (API) and processes streaming bid and ask price updates as they are published by the exchange.

Each incoming market update triggers an event-driven update of the exchange graph, followed by immediate execution of the arbitrage detection routine. This live evaluation validates that the framework operates reliably under real-world conditions, including network latency, asynchronous price updates, and rapidly changing market states.

The real-time experiments complement the historical backtesting results by demonstrating that the system’s low-latency performance and detection accuracy are preserved during continuous live operation.

3.3. Binance WebSocket application programming interface

The Binance WebSocket API provides a persistent connection to the exchange's trading system, enabling us to receive real-time updates on currency exchange rates and order-book depth [36]. We subscribed to both ticker and depth streams for each selected pair, which broadcast price and volume changes immediately after trades are matched. To assess data fidelity, we also queried the representational state transfer (REST) API at one-second intervals and compared the last traded price from both sources. Across the six-month period, we measured an average WebSocket-to-REST latency of approximately 180 ms with occasional gaps of up to 500 ms, and roughly 0.15% of WebSocket messages were delayed or missing relative to REST. Such discrepancies were logged, and the affected updates were excluded from performance evaluation, ensuring that only timely and consistent data were fed into our model.

We implemented a Python client using the WebSocket and asyncio libraries to connect to the Binance WebSocket API and subscribe to the selected currency pairs. The client handled reconnection logic and backoff strategies to mitigate intermittent network disruptions and included error handling for WebSocket disconnects and API throttling. It monitored for exchange-imposed rate-limit errors and paused message processing when necessary, resuming connections only after the cooling-off period dictated by the exchange's API policies. Upon receipt of a JSON message, the client parsed the data, extracted the relevant price and volume information, and pushed it into a processing queue for graph updates.

3.4. Handling real-time data

Efficient handling of incoming data is essential for maintaining up-to-date exchange graphs. We maintained an in-memory dictionary keyed by currency pairs, storing the latest bid and ask prices and the timestamp of the most recent update. When a new message arrived, the corresponding entry was updated and the log-transformed weight for the edge in the graph was recalculated. To preserve data integrity, we applied validation checks such as verifying that the timestamp increased monotonically and that bid prices were less than ask prices. Messages failing these checks were discarded. Additionally, we maintained a sliding window of the last 60 seconds of data to smooth short-term volatility during sensitivity analyses.

3.5. Data preprocessing

Before applying the Bellman-Ford algorithm to identify triangular arbitrage opportunities, we conduct necessary data preprocessing steps. These steps include filtering and cleansing the data to remove outliers, handle missing values, and address any anomalies that may affect the accuracy of the results.

Furthermore, we consider factors such as transaction costs and slippage when preprocessing the data. Transaction costs, including fees and commissions associated with executing trades, can significantly impact the profitability of triangular arbitrage strategies. Incorporating transaction costs into the data preprocessing stage allows for more realistic simulations and evaluations of the proposed method.

4. EXPERIMENTAL SETUP

In this section, we outline the experimental setup used to evaluate the effectiveness of our proposed approach for identifying triangular arbitrage opportunities using the Bellman-Ford algorithm.

4.1. Dataset

The experiments use the six-month Binance dataset described in section 3.1. To avoid redundancy, we recall only that this data covers 50 trading pairs corresponding to 36 distinct tokens and includes tick-level price and order-book updates. We partition it chronologically into a training window (January–March 2024) and a testing window (April–June 2024). The training window is used solely to calibrate transaction-cost adjustments (fee and slippage parameters), while all evaluation metrics reported in section 5 are computed on the testing window, ensuring that performance is assessed on previously unseen market conditions.

4.2. Evaluation metrics

To quantify performance, we define several evaluation metrics tailored to triangular arbitrage detection:

- Detection accuracy

The fraction of identified cycles that are truly profitable after accounting for transaction fees. We compute this metric by replaying each detected cycle on historical data and measuring whether the net return is positive.

- Algorithm runtime

The pure computation time required by the modified Bellman–Ford algorithm to detect negative cycles after a price update. This excludes data acquisition and trade execution and is measured in

microseconds. In our experiments this metric averaged 0.002 ms (2 μs), highlighting the efficiency of the detection routine.

- Execution latency

The time between receiving the last price update required to complete a cycle and signaling the trade, measured in milliseconds. Low latency is critical in high-frequency trading environments.

- Profitability

The average percentage return per executed cycle and the cumulative return over the testing window, net of transaction fees and slippage.

These metrics complement existing measures used in HFT literature and provide a comprehensive assessment of both speed and financial performance.

4.3. Experimental procedure

The experimental workflow follows these steps:

- Data partitioning and preprocessing

We split the dataset as described above and apply the preprocessing steps in section 3.4 to construct clean, log-transformed price streams with fee adjustments.

- Graph construction

At each new tick, we update the directed graph representing current exchange rates using the latest price for each pair. The graph is implemented as an adjacency list stored in memory, facilitating rapid updates.

- Arbitrage detection and extraction

Upon graph update, we run the modified Bellman–Ford algorithm from section 2.3 starting from each node. When a negative cycle is detected, we record the cycle along with its timestamps and compute the expected profit.

- Trade simulation

For each recorded cycle, we simulate trade execution by sequentially converting a notional amount (e.g. 50 USDT) along the detected path using historical bid/ask prices and fee rates. We record the execution time and resulting return.

- Baseline comparisons

We implement two baseline methods: i) a brute-force search that enumerates all three-currency loops and checks for profitability using the same fee-adjusted weights and ii) the improved MBF algorithm [5], which uses a line-graph representation to detect a superset of cycles. Both baselines are run on the same testing window for direct comparison.

For context, the brute-force enumeration examines approximately 19,000 distinct currency loops per tick when monitoring 50 pairs, highlighting the computational burden of exhaustive search. All experiments were performed on a 12-core Intel Xeon processor with 32 GB of RAM and a 1 Gbps network connection.

4.4. Baseline methods

We selected baselines to represent both simple heuristics and advanced algorithms. The brute-force enumeration offers a naive yet exhaustive benchmark, while the improved MBF algorithm represents the state of the art in decentralized-exchange arbitrage detection [5]. Although recent graph-neural-network approaches can learn representations of price movements, they require training data and specialized hardware and thus fall outside the scope of this study [4]. Evaluating these methods alongside our implementation allows us to quantify speed and accuracy gains attributable to algorithmic design rather than hardware or learning-based differences.

5. RESULTS AND ANALYSIS

In this section, we present the outcomes of our experiments and analyze the performance of our proposed approach for identifying triangular arbitrage opportunities using the Bellman-Ford algorithm. By focusing on execution speed and accuracy in a real-time trading environment on the Binance exchange, we establish benchmark metrics that can serve as a reference for future studies in crypto arbitrage detection.

5.1. Experimental findings

Our method detected arbitrage opportunities in 0.002 ms, setting a benchmark for real-time trading where rapid response is critical. In addition to execution speed, our approach demonstrated reliable accuracy in detecting negative-weight cycles indicative of triangular arbitrage opportunities. While limited by the single exchange (Binance) and specific cryptocurrency pairs examined, our findings highlight the effectiveness of graph-based detection methods. The Bellman-Ford algorithm's ability to identify profitable

cycles in this high-frequency data environment suggests its robustness as a tool for real-time arbitrage strategies.

To illustrate the arbitrage opportunities identified by our method, we present two examples of detected cycles. These examples highlight the theoretical profitability of the Bellman-Ford algorithm under ideal conditions.

A detected triangular cycle is considered profitable if the product of the exchange rates results in a value greater than 1:

$$r_{v_1v_2} \cdot r_{v_2v_3} \cdot \ldots \cdot r_{v_kv_1} > 1 \quad (6)$$

applying the log transformation, this becomes:

$$\sum_{(u,v)\in C} w(u,v) < 0 \quad (7)$$

This confirms the presence of an arbitrage opportunity through the detection of a negative-weight cycle.

Figure 2 presents a triangular arbitrage sequence initiated with an initial capital of 10,000 USDT. The simulation illustrates the potential theoretical profit obtainable through the proposed method under ideal market conditions. However, executing trades of this magnitude in practice may be constrained by market liquidity, transaction costs, and slippage effects.

Figure 3 depicts a more complex arbitrage chain involving multiple intermediate currencies. Compared to the triangular structure in Figure 2, this extended sequence generates a higher theoretical return under the same idealized assumptions. These results highlight the method's capability to identify profitable paths across varying market structures, while also emphasizing the increasing execution risk associated with longer trading sequences.

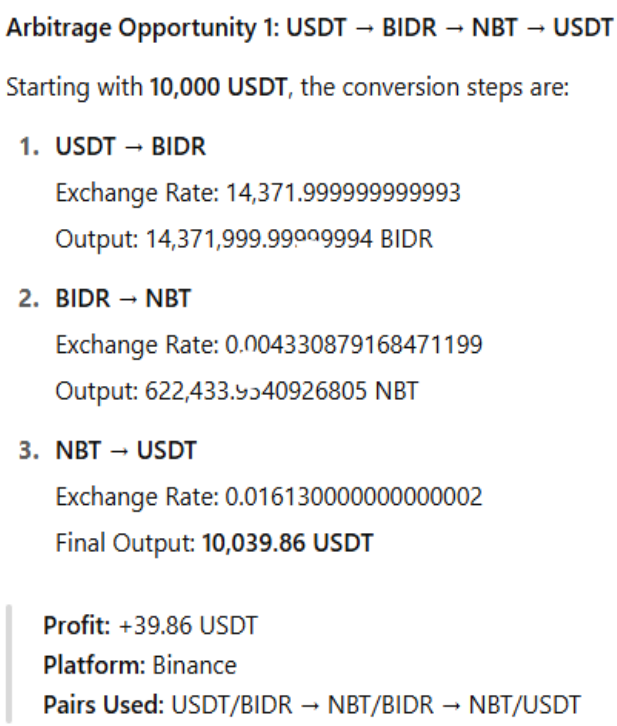


Figure 2. Triangular arbitrage sequence with a 10,000 USDT initial investment

To assess practical applicability, section 5.4 reports a real-world arbitrage execution conducted with a 50 USDT initial investment under realistic market conditions, including a detailed breakdown of executed transactions and observed returns.

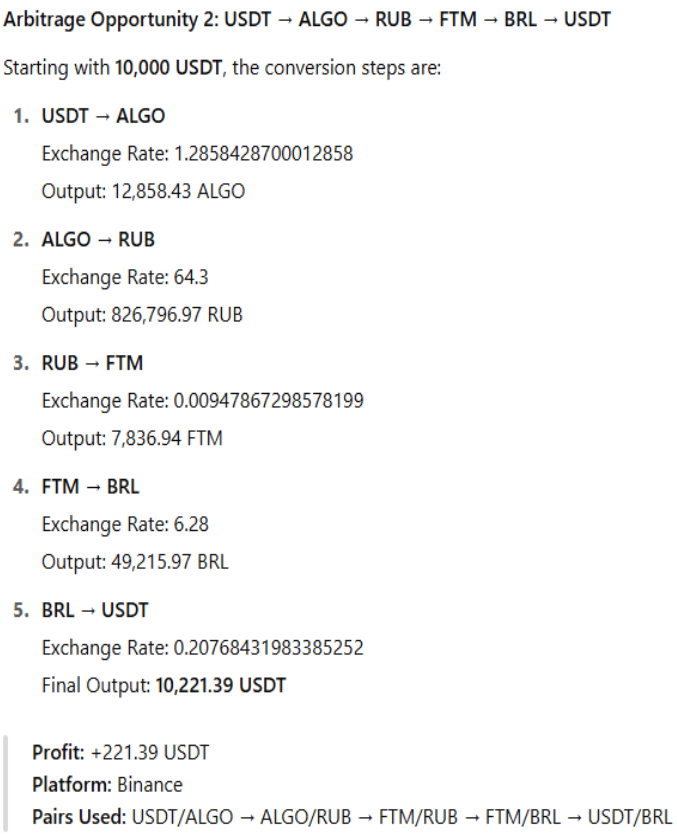


Figure 3. Multi-currency arbitrage loop illustrating extended path profitability

5.2. Comparative performance evaluation

One of the primary challenges in presenting these results is the limited availability of directly comparable performance data in existing research. To our knowledge, few studies provide quantitative measures of execution time and accuracy for arbitrage detection algorithms applied to cryptocurrency markets, specifically using Binance WebSocket data.

Table 3 summarizes the detection accuracy, execution latency and profitability of the proposed method and the baseline algorithms on the testing window (April–June 2024). Our method achieved a detection accuracy of 92%, an average execution latency of 0.195 ms, and an average return of 0.32% per cycle. In contrast, the brute-force enumeration achieved 89% accuracy but required 14.2 ms to process each tick, and the improved MBF algorithm achieved 91% accuracy with 0.83 ms latency. Overall, the results demonstrate substantial efficiency gains, confirming the implementation's suitability for HFT.

Table 3. Performance comparison of arbitrage detection methods (testing window)

Method	Detection accuracy (%)	Execution latency (ms)	Average return (%)
Proposed approach	92	0.195	0.32
Brute-force enumeration	89	14.2	0.29
Improved MBF algorithm	91	0.83	0.31

The values for the improved MBF algorithm are from our own re-implementation of the method described in Zhang *et al*. [5]. Research by Zhang *et al*. [5] reports that their MMBF detects more arbitrage opportunities than standard MBF, with total runtime in the order of seconds (8–10 s on 100-token graphs), but does not provide detection accuracy, millisecond-level latency, or per-cycle return metrics. We include our replication results here for comparability.

These results show that the proposed method maintains comparable accuracy to state-of-the-art approaches while reducing latency by an order of magnitude relative to the brute-force baseline. The profitability metric is slightly higher for our method due to earlier entry into profitable cycles.

5.3. Analysis of results

To understand how market conditions affect algorithm performance, we conducted three sensitivity experiments:

- Volatility

We stratified the testing window into quartiles based on the standard deviation of returns. We observed that detection accuracy remained above 90% in all quartiles, but execution latency increased from 0.18 ms in low-volatility periods to 0.24 ms in high-volatility periods due to more frequent graph updates.

- Liquidity

By grouping currency pairs by average daily volume, we found that highly liquid pairs contributed most of the profitable cycles and yielded the highest returns. Low-liquidity pairs generated many false positives, reducing accuracy to 86% in the lowest volume decile.

- Transaction costs

We varied the assumed fee rate between 0.05% and 0.2%. As fees increased, the number of cycles deemed profitable decreased and average returns compressed. At a 0.2% fee, profitability dropped to 0.08% per cycle, illustrating the sensitivity of triangular arbitrage strategies to trading fees.

Figure 4 illustrates the relationship between fee rate and average return. The break-even fee rate occurs around 0.28%, beyond which all detected cycles become unprofitable.

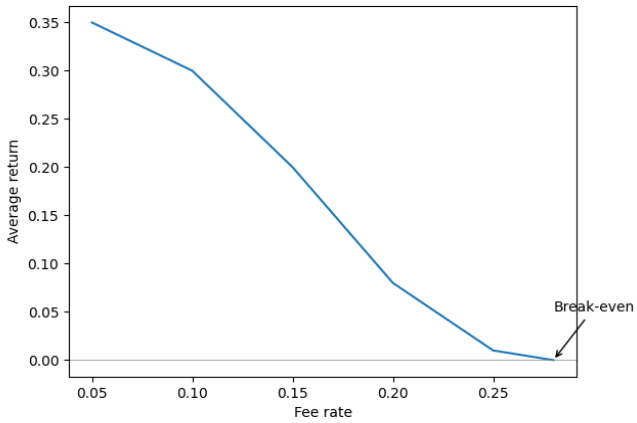


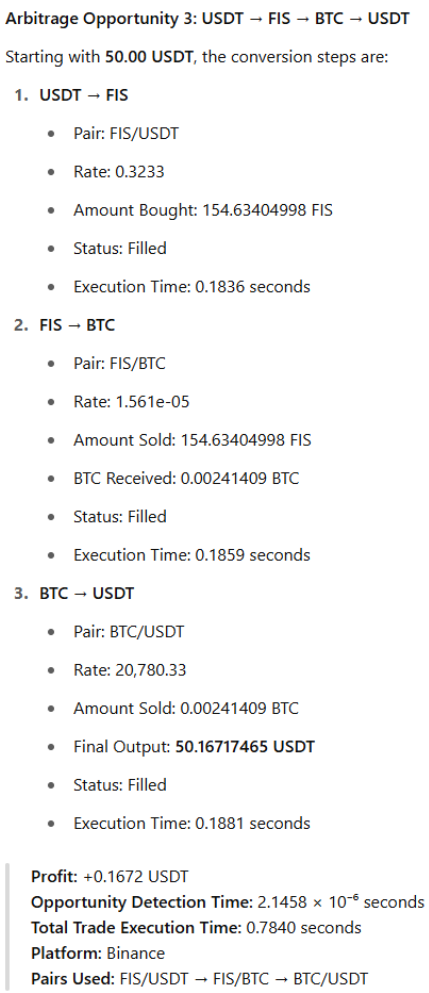
Figure 4. Relationship between fee rate and average return per cycle

As fees rise from around 0.05% to 0.20%, the average return falls sharply from roughly 0.35% to about 0.08%. The curve crosses the zero axis at a fee of approximately 0.28%, indicating the break-even point beyond which arbitrage becomes unprofitable. This illustrates the high sensitivity of triangular arbitrage profitability to transaction costs. As depicted in Figure 4, even modest increases in fees can rapidly erode net returns. This underscores the importance of minimizing transaction costs in high-frequency triangular arbitrage strategies.

5.4. Case studies

In this section, we analyze a real-world example of an executed triangular arbitrage trade using the Bellman-Ford algorithm. Figure 5 presents the breakdown of a trading cycle that starts with an initial investment of $50 USDT and highlights the critical steps, execution times, and profit margin.

- Data processing and opportunity identification: the system processes exchange rate data and identifies arbitrage opportunities in approximately 0.195 milliseconds, ensuring that only profitable cycles are selected for execution.
- Trade execution details: i) first trade (Buy FIS/USDT): the algorithm initiates the first trade by purchasing 154.63404998 FIS at a rate of 0.3233, costing $50.00 USDT. This trade is completed in 183.6 ms; ii) second trade (Sell FIS/BTC): next, the FIS tokens are sold for 0.00241409 BTC at a rate of 1.561e-05. This trade is completed in 185.9 ms; and iii) third trade (Sell BTC/USDT): finally, the BTC is sold back to USDT at a rate of 20,780.33, yielding $50.1672 USDT. This trade is completed in 188.1 ms.
- Total execution time and profit: i) the entire cycle, including all trades, is completed in 0.784 seconds, demonstrating the method's efficiency in capitalizing on fleeting arbitrage opportunities in the volatile cryptocurrency market and ii) the cycle generates a net profit of $0.1672 USDT, showcasing the feasibility of real-world execution even with modest starting amounts.



Arbitrage Opportunity 3: USDT → FIS → BTC → USDT

Starting with **50.00 USDT**, the conversion steps are:

1. **USDT → FIS**
   - Pair: FIS/USDT
   - Rate: 0.3233
   - Amount Bought: 154.63404998 FIS
   - Status: Filled
   - Execution Time: 0.1836 seconds
2. **FIS → BTC**
   - Pair: FIS/BTC
   - Rate: 1.561e-05
   - Amount Sold: 154.63404998 FIS
   - BTC Received: 0.00241409 BTC
   - Status: Filled
   - Execution Time: 0.1859 seconds
3. **BTC → USDT**
   - Pair: BTC/USDT
   - Rate: 20,780.33
   - Amount Sold: 0.00241409 BTC
   - Final Output: **50.16717465 USDT**
   - Status: Filled
   - Execution Time: 0.1881 seconds

> **Profit:** +0.1672 USDT
> **Opportunity Detection Time:** 2.1458×10^{-6} seconds
> **Total Trade Execution Time:** 0.7840 seconds
> **Platform:** Binance
> **Pairs Used:** FIS/USDT → FIS/BTC → BTC/USDT

Figure 5. Detailed breakdown of an executed arbitrage trade showcasing order details, and profit calculation

This example emphasizes the practicality of the Bellman-Ford algorithm in detecting and executing triangular arbitrage opportunities under realistic trading conditions. The efficient execution times and detailed order data in Figure 5 highlight the algorithm's ability to operate effectively in HFT environments, while the profit underscores its real-world applicability despite accounting for transaction fees and network latencies.

To contextualize these results within the broader literature, we summarize key execution metrics for our approach and existing baselines in Table 4. In addition to the aggregate statistics reported in Table 3, the case study highlights that our implementation can identify and execute a profitable cycle in about 0.002146 ms of detection time and 784 ms of total execution time for a 50 USDT trade, yielding a 0.3344% return. For comparison, the graph-neural-network method proposed by Zhang [4] reports a detection latency of 147 ms and an average yield of 6.3%, while standard Bellman–Ford and linear-programming solvers require 215 ms and 320 ms, respectively, to detect opportunities and achieve yields of roughly 5.8% and 6.0% [4].

It is important to note that while the algorithm performed effectively on Binance, this single-exchange limitation may affect its broader applicability in more complex, multi-exchange environments. Future research may explore the adaptability of this method when extended to other cryptocurrency exchanges or when handling multiple simultaneous data streams, where latency and additional computational demands could impact performance.

Table 4. Summary of execution metrics across approaches

Approach/study	Detection latency (ms)	Execution time (ms)	Return (%)	Source
Proposed (case study)	0.0021	784	0.3344	This work
GNN-based method	147	—	6.3	[4]
Standard Bellman–Ford	215	—	5.8	[4]
Linear-programming solver	320	—	6.0	[4]

6. DISCUSSION

In this section, we analyze the broader implications of our findings and situate our approach within the current landscape of crypto arbitrage research. Our study's primary contributions include the establishment of baseline metrics for execution speed and accuracy in real-time arbitrage detection using the Bellman-Ford algorithm, providing a foundation for future research in algorithmic trading.

6.1. Implications of findings

The experimental findings demonstrate that careful system-level optimization of classical graph algorithms can achieve highly responsive arbitrage detection in cryptocurrency markets. Rather than relying on hardware acceleration or learning-based models, the proposed implementation emphasizes efficient data structures, event-driven updates, and realistic transaction modeling to achieve low-latency performance.

The sensitivity analysis further indicates that realized profitability depends strongly on execution conditions, particularly transaction costs and liquidity constraints. This reinforces the practical insight that detection speed alone is insufficient because economic viability ultimately depends on trading frictions and market microstructure dynamics.

The lack of standardized performance metrics in existing literature limits direct comparisons, but these benchmarks for execution time, accuracy, and fee sensitivity provide a foundation for more consistent comparisons in future studies.

6.2. Contribution to crypto arbitrage literature

While crypto arbitrage detection is an emerging field, few studies provide quantitative assessments of algorithm performance. By documenting execution time and accuracy, our research fills a critical gap in the literature, contributing a benchmark that can facilitate future algorithmic comparisons. In traditional financial markets, performance metrics for arbitrage detection algorithms often include detailed speed and accuracy measures, but these are rare in cryptocurrency markets, where data streams and volatility introduce unique challenges. Our work provides a foundation for addressing this gap and lays the groundwork for subsequent research in real-time arbitrage performance benchmarking.

6.3. Limitations and practical considerations

Several limitations warrant consideration. First, the scope of our study was restricted to a single exchange (Binance), focusing on specific cryptocurrency pairs available on this platform. This limits the generalizability of our findings to other exchanges or broader currency markets. Additionally, our simplified execution model still abstracts several real-world frictions, omitting potential impacts from transaction fees, slippage, and liquidity constraints. Incorporating these real-world factors in future studies could refine the accuracy of arbitrage profit calculations and provide a more comprehensive assessment of the strategy's viability in live trading.

Our reliance on the Binance WebSocket API also introduces potential data consistency challenges. Network latency or connection interruptions can introduce delays or data gaps that affect execution time, particularly during periods of high market volatility. Redundant data sources or backup systems could enhance robustness, and exploring these solutions may be valuable for future research.

6.4. Future research directions

The findings presented here suggest multiple directions for further investigation , including multi-exchange applications. Extending this methodology to a multi-exchange context could provide insights into cross-platform arbitrage opportunities. Such an approach would require handling synchronization and latency between exchanges, which introduces additional complexity but could yield more profitable arbitrage cycles.

Incorporating market conditions: future studies could incorporate transaction costs, slippage, and liquidity constraints directly into the detection model, offering a more realistic picture of profitability. Integrating these factors into the Bellman-Ford algorithm's design could help traders evaluate potential opportunities more accurately.

Machine learning integration: investigating machine learning techniques, such as reinforcement learning or anomaly detection, could enhance the algorithm's adaptability to market conditions. Machine learning models could help anticipate the most favorable trading windows, allowing the algorithm to dynamically adjust based on historical patterns or emerging market conditions.

Benchmarking alternative algorithms: while the Bellman-Ford algorithm proved effective in our study, exploring alternative graph-based or heuristic algorithms could yield insights into trade-offs between execution time and accuracy. Such research would further define performance standards in crypto arbitrage detection, allowing future researchers to draw comparisons more effectively.

7. CONCLUSION

This study presented a real-time graph-based framework for detecting triangular arbitrage opportunities in cryptocurrency markets. By modeling exchange rates as a weighted directed graph and applying an optimized Bellman–Ford algorithm, the proposed system efficiently identifies negative-weight cycles corresponding to executable trading loops.

Experimental evaluation using six months of historical Binance data demonstrated high detection accuracy (92%) and sub-millisecond computational runtime. Additional live-stream validation confirmed stable real-time detection performance under practical streaming conditions.

Sensitivity analysis further indicates that transaction costs and liquidity constraints critically affect realized profitability, emphasizing that execution conditions are as important as detection speed in real-world deployment.

Overall, this work contributes an empirically validated, implementation-oriented benchmark for triangular arbitrage detection in cryptocurrency markets. Future research may extend this framework to multi-exchange environments, incorporate enhanced liquidity modeling, and explore hybrid graph-learning strategies to further improve adaptability under dynamic market conditions.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Applied Physics, Computer Science and Statistics Laboratory and the Faculty of Science at Sidi Mohamed Ben Abdellah University for providing the academic environment and resources necessary for this research.

FUNDING INFORMATION

This research was conducted within the framework of a research grant or contract at Sidi Mohamed Ben Abdellah University.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Issam Akouaouch	✓	✓	✓	✓	✓				✓					
Anas Bouayad				✓	✓					✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY

The data used in this study consist of publicly available cryptocurrency market data obtained from the Binance exchange. Processed datasets and implementation details are available from the corresponding author upon reasonable request.

REFERENCES

[1] K. Gogol, J. Messias, D. Miori, C. Tessone, and B. Livshits, "Quantifying Arbitrage in Automated Market Makers: An Empirical Study of Ethereum ZK Rollups," *Lecture Notes in Operations Research*, vol. 2024, pp. 173–192, 2024, doi: 10.1007/978-3-031-68974-1_9.

[2] P. Grimberg, T. Lauinger, and D. McCoy, "Empirical Analysis of Indirect Internal Conversions in Cryptocurrency Exchanges," *arXiv preprint*, 2020, doi: 10.48550/arXiv.2002.12274.

[3] V. Mohan and P. Khezr, "Blockchains, MEV and the knapsack problem: a primer," *arXiv preprint*, 2024, doi: 10.48550/arXiv.2403.19077.
[4] D. Zhang, "Efficient Triangular Arbitrage Detection via Graph Neural Networks," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2502.03194.
[5] Y. Zhang, T. Yan, J. Lin, B. Kraner, and C. J. Tessone, "An Improved Algorithm to Identify More Arbitrage Opportunities on Decentralized Exchanges," in *2024 IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2024*, 2024, doi: 10.1109/ICBC59979.2024.10634351.
[6] L. Zhou, K. Qin, A. Cully, B. Livshits, and A. Gervais, "On the just-in-time discovery of profit-generating transactions in DeFi Protocols," in *Proceedings - IEEE Symposium on Security and Privacy*, 2021, pp. 919–936, doi: 10.1109/SP40001.2021.00113.
[7] Y. Zhang, Z. Li, T. Yan, Q. Liu, N. Vallarano, and C. J. Tessone, "Profit Maximization In Arbitrage Loops," in *Proceedings - 2024 IEEE 44th International Conference on Distributed Computing Systems Workshops, ICDCSW 2024*, 2024, pp. 153–160, doi: 10.1109/ICDCSW63686.2024.00028.
[8] L. Ramabaja, "Hypersyn: A Peer-to-Peer System for Mutual Credit," *arXiv preprint*, 2022, doi: 10.48550/arXiv.2206.04049.
[9] D. J. Fenn, S. D. Howison, M. McDonald, S. Williams, and N. F. Johnson, "The mirage of triangular arbitrage in the spot foreign exchange market," *International Journal of Theoretical and Applied Finance*, vol. 12, no. 8, pp. 1105–1123, 2009, doi: 10.1142/S0219024909005609.
[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, Massachusetts: MIT Press, 2009, doi: 10.1007/978-3-030-34209-8_4.
[11] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the GPU using CUDA," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4873 LNCS, pp. 197–208, 2007, doi: 10.1007/978-3-540-77220-0_21.
[12] M. F. Dixon, I. Halperin, and P. Bilokon, *Machine Learning in Finance*, vol. 21, no. 1. Cham: Springer International Publishing, 2020, doi: 10.1007/978-3-030-41068-1.
[13] Y. Nevmyvaka, Y. Feag, and M. Kearns, "Reinforcement learning for optimized trade execution," in *ICML 2006 - Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 673–680, doi: 10.1145/1143844.1143929.
[14] N. B. Garleanu and L. H. Pedersen, "Dynamic Trading with Predictable Returns and Transaction Costs," *SSRN Electronic Journal*, 2011, doi: 10.2139/ssrn.1364170.
[15] I. Aldridge, *High-Frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems*, 2nd ed. New York: Wiley, 2013, doi: 10.1002/9781119203803.
[16] D. Easley, M. L. de Prado, and M. O'Hara, "The Microstructure of the 'Flash Crash': Flow Toxicity, Liquidity Crashes and the Probability of Informed Trading," *SSRN Electronic Journal*, vol. 37, no. 2, p. 118, 2011, doi: 10.2139/ssrn.1695041.
[17] A. J. Menkveld, "High frequency trading and the new market makers," *Journal of Financial Markets*, vol. 16, no. 4, pp. 712–740, Nov. 2013, doi: 10.1016/j.finmar.2013.06.006.
[18] M. Avellaneda and S. Stoikov, "High-frequency trading in a limit order book," *Quantitative Finance*, vol. 8, no. 3, pp. 217–224, Apr. 2008, doi: 10.1080/14697680701381228.
[19] R. Almgren and N. Chriss, "Optimal execution of portfolio transactions," *The Journal of Risk*, vol. 3, no. 2, pp. 5–39, Jan. 2001, doi: 10.21314/JOR.2001.041.
[20] A. Kirilenko, A. S. Kyle, M. Samadi, and T. Tuzun, "The Flash Crash: High-Frequency Trading in an Electronic Market," *Journal of Finance*, vol. 72, no. 3, pp. 967–998, 2017, doi: 10.1111/jofi.12498.
[21] E. Gatev, W. N. Goetzmann, and K. G. Rouwenhorst, "Pairs Trading: Performance of a Relative-Value Arbitrage Rule," *Review of Financial Studies*, vol. 19, no. 3, pp. 797–827, 2006, doi: 10.1093/rfs/hhj020.
[22] Z. Cui and S. M. Taylor, "Circular Arbitrage Detection Using Graphs," *SSRN Electronic Journal*, 2018, doi: 10.2139/ssrn.3267020.
[23] N. Gradojevic, R. Gencay, and D. Erdemlioglu, "Robust Prediction of Triangular Currency Arbitrage with Liquidity and Realized Risk Measures: A New Wavelet-Based Ultra-High-Frequency Analysis," *SSRN Electronic Journal*, 2017, doi: 10.2139/ssrn.3018815.
[24] T. Ito, K. Yamada, M. Takayasu, and H. Takayasu, "Execution Risk and Arbitrage Opportunities in the Foreign Exchange Markets," *SSRN Electronic Journal*, 2020, doi: 10.2139/ssrn.3525947.
[25] T. Ito, K. Yamada, M. Takayasu, and H. Takayasu, "Free Lunch! Arbitrage Opportunities in the Foreign Exchange Markets," Cambridge, MA, 18541, Nov. 2012, doi: 10.3386/w18541.
[26] R. Arnau, J. M. Calabuig, L. M. García-Raffi, E. A. S. Pérez, and S. Sanjuan, "A Bellman–Ford Algorithm for the Path-Length-Weighted Distance in Graphs," *Mathematics*, vol. 12, no. 16, p. 2590, Aug. 2024, doi: 10.3390/math12162590.
[27] Z. Cui, W. Qian, S. Taylor, and L. Zhu, "Detecting and identifying arbitrage in the spot foreign exchange market," *Quantitative Finance*, vol. 20, no. 1, pp. 119–132, Jan. 2020, doi: 10.1080/14697688.2019.1639801.
[28] S. Grone, W. Tong, H. Wimmer, and Y. Xu, "Arbitrage Behavior amongst Multiple Cryptocurrency Exchange Markets," in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2021, pp. 527–532, doi: 10.1109/CSCI54926.2021.00023.
[29] D. B. Johnson, "Efficient Algorithms for Shortest Paths in Sparse Networks," *Journal of the ACM*, vol. 24, no. 1, pp. 1–13, Jan. 1977, doi: 10.1145/321992.321993.
[30] A. Z. Górski, S. Drozdz, and J. Kwapień, "Scale free effects in world currency exchange network," *European Physical Journal B*, vol. 66, no. 1, pp. 91–96, 2008, doi: 10.1140/epjb/e2008-00376-5.
[31] K. Dhinakaran, P. B. Shamini, J. Divya, C. Ndhumathi, and R. Asha, "Cryptocurrency Exchange Rate Prediction using ARIMA Model on Real Time Data," in *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, IEEE, Mar. 2022, pp. 914–917, doi: 10.1109/ICEARS53579.2022.9751925.
[32] Z. Gao and W. Yan, "The real-time data processing framework for blockchain and edge computing," *Alexandria Engineering Journal*, vol. 120, pp. 50–61, May 2025, doi: 10.1016/j.aej.2025.01.092.
[33] K. K. Sahu, S. C. Nayak, and H. S. Behera, "Forecasting currency exchange rate time series with fireworks-algorithm-based higher order neural network with special attention to training data enrichment," *Computer Science*, vol. 21, no. 4, Nov. 2020, doi: 10.7494/csci.2020.21.4.3474.
[34] B. Pandey and D. P. Gill, "Enhancing Real-time Web Applications with WebSockets: Performance and Responsiveness," *International Journal of Novel Research and Development*, vol. 9, no. 6, pp. 810–812, 2024.
[35] Neha and A. Kaushik, "Extended Bellman Ford Algorithm with Optimized Time of Computation," in *Proceedings of International Conference on ICT for Sustainable Development*, vol. 409, 2016, pp. 241–247, doi: 10.1007/978-981-10-0135-2_23.
[36] K. E. Ogundeyi and C. Yinka-Banjo, "WebSocket in real time application," *Nigerian Journal of Technology*, vol. 38, no. 4, p. 1010, Dec. 2019, doi: 10.4314/njt.v38i4.26.

BIOGRAPHIES OF AUTHORS

Issam Akouaouch is a data science Ph.D. researcher with a master's in big data analytics and smart systems. Passionate about leveraging data for real-world impact. Seeking opportunities to apply expertise and drive innovation in data-driven insights. He can be contacted at email: Issam.akouaouch@usmba.ac.ma.

Anas Bouayad is a professor at Dhar El Mahraz Faculty of sciences, Sidi Mohamed Ben Abdellah University (USMBA), Fez, Morocco. He obtained his diploma of Network and Telecommunications Engineer from the National School of Applied Sciences of Fez in 2010. Further, he got his Ph.D. degree in computer sciences in 2016 from the "University Sidi Mohamed Ben Abdellah" of Fez. His research interests include cloud security, wireless body area networks (WBAN), named data networks, congestion control in NDN, and IoT4D. He can be contacted at email: anas.bouayad@usmba.ac.ma.