Arbitrage trading between decentral and central cryptocurrency exchanges

Lennart Schwertfeger[†‡] Bodo Vogt[†]

This draft: January 13, 2025

Abstract

This paper demonstrates practical arbitrage trading on the cryptocurrency market. It provides guidance on how to build a high-frequency trading system that benefits from exhibiting arbitrage opportunities. It reveals the algorithm of the trading bot that incorporates the order placement and execution strategy between decentral and central cryptocurrency exchanges. The arbitrage algorithm is implemented on two different blockchains that interact with Uniswap and Balancer folks. Current research explores arbitrage opportunities with back-testing models, the paper focuses on trades with realized arbitrage trades. Practical arbitrage includes all operational costs, liquidity constraints, direct effects on markets, and competition with peer arbitrage traders.

JEL Classification: G14, G15, G17, G19

Keywords: high-frequency trading, distributed ledger technology, smart contract, arbitrage system

[†]Department of Empirical Economics, Otto-von-Guericke University Magdeburg, Germany
[‡]Corresponding author, email lennart.schwertfeger@ovgu.de

1 Introduction and literature review

According to DefiLlama,[1] over 250 blockchains exist, which provide distinct services, use different security and consensus mechanisms, and attract different groups of users. The trilemma between security, scalability, and decentralization allows user to place their assets on various blockchains, taking advantage depending on their individual preferences. However, the network communication between blockchains remains weak and demands solutions to improve the connection across various blockchains. Jin et al. (2018) write about interoperability between blockchains and indicate the advantages of improved connectivity, which enhances smart contract service. A major problem that occurs through weak interoperability is a fragmented capital system. Whereas most blockchains are decentralized, they often require central third parties to bridge assets between them. Caldarelli (2021) provides an overview of wrapped token issuing procedures, i.e., bridging. This process includes locking the original asset on its native blockchain and minting a token copy on another chain. While bridging improves connections between blockchains, it results in additional costs for users, i.e., wrapping costs.

Centralized exchanges (CEXs) can be seen as another intermediary since they facilitate liquidity transfer between blockchains. They play a crucial role in the blockchain ecosystem by allowing users to deposit funds from one blockchain, trade assets within the exchange's platform, convert one asset to another, and withdraw the converted funds to a different blockchain. Through this process, CEXs provide a mechanism for cross-chain liquidity transfers. In contrast to CEXs, where users trust third-party custodians, decentralized exchanges (DEXs) are protocols that include various smart contracts deployed directly on the blockchain. Jiao et al. (2024) define smart contracts as computerized transaction agreements that serve as a trusted intermediary. DEX smart contracts enable the peer-to-peer exchange of assets without an intermediary and ensure that the blockchain's consensus rather than centralized entities govern transactions. Various forms of DEXs exist that support peer-to-peer trading on the blockchain. Unlike limit order book systems, known from traditional financial markets and CEXs, most DEXs rely on automated market makers (AMMs). Xu et al. (2023) present a comprehensive analysis of various AMM smart contracts. They categorize contracts based on their conversation functions, i.e., the mathematical pricing model behind AMMs, asset provisioning mechanisms, exchange rate determination, and slippage calculations. It requires various conversation functions to allow efficient trading for specific trading pairs. For example, Balancer,[2] originally launched on the Ethereum[3] blockchain, offers two specific trading pools: weighted pools and composable stable pools. Weighted pools are designed for trading assets with volatile prices. In contrast, stable pools are optimized for trading assets that maintain stable prices, such as fiat currencies, e.g., USDC and USDT, which aim to peg to the value of the U.S. dollar. Despite the adjustment of conversation functions towards specific trading pool needs, other DEX innovations, such as smart order routing, have improved the efficiency of DEX trading by automating price comparisons among trading pools. Before these innovations, DEX traders had to manually compare prices across different trading pools to achieve the best possible execution price. Smart order routing selects the most efficient route for trade and can also split trades across multiple routes to further enhance price efficiency. While these innovations improved trading on individual exchanges, DEX aggregators have expanded this optimization by improving order flow across multiple exchanges on a single blockchain. Similar to smart order routing, DEX aggregators like Jupiter[4] (on the Solana[5] blockchain) and 1inch[6] (on the Ethereum blockchain) compare prices across various exchanges and liquidity pools to optimize order flow for traders. The improved order flows for blockchain trading activities lead to reduced price

[1] See official website.
[2] See Martinell and Mushegian (2019).
[3] See Buterin (2014).
[4] See JupiterChain (2018).
[5] See Yakovenko (2017).
[6] See Bukov et al. (2024).

differences along exchanges and trading pairs. The bypassing effect is a reduced amount of arbitrage opportunities for arbitrage traders. Arbitrage traders benefit from selling and buying an equivalent amount of an asset at the same time, as they take advantage of price differences. Arbitrage trader that interacts with AMM DEX, benefit from the static nature of the protocol. If the market price of an asset moves, arbitrage traders interact with the DEX to benefit from the old market prices. This happens at the cost of liquidity providers that fund the liquidity pool. Whereas liquidity providers profit from trading fees that arise through trading activities, they can also suffer a financial loss, so-called divergence loss (impermanent loss). Divergence loss is the loss that results from changes in the exposure position of the liquidity provider. Through trading activities, the pool reserves and, correspondingly, the exposure position for liquidity providers change. The divergence loss effect leads to an increased exposure position of the under-performing and a decreased position of the over-performing asset. Loesch et al. (2021) investigate the behavior of liquidity providers at the Uniswap[7] DEX for the time horizon from May 5, 2021, to September 20, 2021. Their analysis shows that liquidity providers generated $199.3 million in fees while they lost $260.1 million due to the divergence loss of their funded assets. As they state, 49.5% of all liquidity providers had negative returns during the analyzed time horizon.

The liquidity provider's financial loss comes to the arbitrage trader's advantage. To measure the potential of arbitrage opportunities, many researchers used historical datasets to back-test arbitrage opportunities between CEXs. Bruzgė and Šapkauskienė (2022) examine Bitcoin arbitrage opportunities across 13 different CEXs using minute-level data from January 2018 to April 2020. Their findings indicate that the network structure of the analyzed exchanges influences these arbitrage opportunities. The authors highlight that price discrepancies persist despite the rapid growth of cryptocurrency markets, enabling profitable arbitrage trading. Makarov and Schoar (2020) demonstrate that price discrepancies for Bitcoin, Ethereum, and Ripple were more pronounced across different countries than within the same country during the period from December 2017 to February 2018. They observe an average arbitrage spread of 15% between cryptocurrency markets in Korea and the US. Building upon their research, Crépellière et al. (2023) extend their analysis to examine how these price differences for Bitcoin, Ethereum, and Ripple evolved over time. They report a continuous decline in price deviations since the first quarter of 2018, with a maximum average price difference of 1.1% after 2019. The research on arbitrage opportunities focuses solely on CEXs and heavily relies on back-testing methods. DEXs are rarely included in such analyses, primarily due to the significant cost and time required to extract high-frequency data from blockchains. Data providers such as Kaiko, Cryptocompare, or similar entities must host a blockchain node to collect the necessary data. Hosting a blockchain node is resource-intensive and time-consuming, as nodes can lose network connectivity, leading to data inconsistencies. Păuna (2018) describes typical issues when retrieving data from cryptocurrency exchanges. The author states that each exchange has its own application programming interface and data structure. Further, exchanges have different interrogation mechanisms, and the allowable number of data requests per time interval differs across platforms. Even with access to high-frequency data, accurately assessing the true potential of arbitrage opportunities remains challenging for researchers, as timestamps across different exchanges are unlikely to align precisely at the millisecond level. The results derived from theoretical back-testing are positively biased, as they presuppose the constant availability of sufficient liquidity to exploit arbitrage opportunities. Further, the theoretical approach overlooks the presence of competition among arbitrageurs and disregards the impact of orders on the markets. Therefore, practical arbitrage opportunities differ from theoretical back-testing, as they account for the actual realized arbitrage spread, which incorporates the impact of price changes on the markets and incorporates arbitrage competition that can lead to unprofitable trades.

[7] See Uniswap (2020).

To track realized arbitrage opportunities, we developed a high-frequency arbitrage trading bot that executes trades between a DEX and a CEX. We have chosen to design the bot for trading between a DEX and a CEX because we argue that arbitrage opportunities between DEXs are diminishing due to on-chain innovations, while opportunities between CEXs are challenging to exploit due to the economies of scale advantages enjoyed by high-volume traders. Schleifer and Vishny (1997) argue that arbitrage trading is primarily carried out by a small group of specialized professionals rather than by many traders. This insight forms a key understanding of arbitrage in traditional financial markets, emphasizing the barriers to entry, such as the requirement for specialized expertise, significant capital, and the ability to execute trades rapidly. Given the experience of developing an algorithm, setting up three secure and efficient arbitrage systems, and 3,621 profitable and 96 unprofitable arbitrage trades, we confirm the statement from Schleifer and Vishny (1997) on the cryptocurrency market. Practical arbitrage between DEX and CEX demands a comprehensive understanding of various components, i.e., smart contracts, blockchain consensus mechanisms, transaction sequencing, cost structures, slippage factors, trading algorithms, and node management. Even minor gaps in this knowledge can result in unprofitable trades, which explains why only a select group of specialized arbitrage traders remain active in these markets. To shed light on the field of practical arbitrage trading, the paper documents all the learning and experiences from cross-exchange arbitrage trading between DEX and CEX.

The rest of the paper is organized as follows: Section 2 introduces mathematical foundations to establish an arbitrage trading system between DEX and CEX. The slippage effect describes the impact of an order on the spot price on the market. Our trading algorithm contains trades at the DEX, limit and market order at the CEX. To avoid uncertainty connected with slippage, subsecion 2.1 describes how to determine ex-ante the expected slippage for the DEX order and CEX market order. Slippage can be seen as a cost of practical trading, as it describes the price difference between the spot and execution price. Subsection 2.2 continues with a profit and cost function that includes all costs when performing practical arbitrage. This subsection also illustrates how the algorithm of an arbitrage system is dependent on the cost structure of the blockchain and exchanges in use. For various cost parameters, trade-offs are discussed that allow the optimization of an arbitrage system. Subsection 2.3 demonstrates that arbitrage traders cannot rely on price data from smart contracts and demonstrates how to incorporate mempool transactions to update DEX price data. Speed is an important factor for successful high-frequency trading at the CEX. For trading at the DEX, the success of trade executions is not purely dependent on speed but also on the consensus mechanism of the blockchain. The consensus of the blockchain decides upon the execution sequence of trades. The manipulation of the execution sequence allows attackers to front-run and sandwich DEX traders. Therefore, subsection 2.3 also describes how arbitrage trader can choose their trading volume to prevent becoming a victim of sandwich attacks. Further, it demonstrates mathematically and with the help of a simulation how the transaction sequence can have an influence on the spot price. The simulation demonstrates that traders can not know the current spot price if the transaction sequence is not known. Section 3 introduces the setup and the algorithm of our arbitrage trading system. Any arbitrage system can be set up on a local computer or on a server. In subsection 3.1, we show the advantages of a server system and explain how to securely set it up. Subsection 3.2 describes our algorithm that can be used for cross-exchange arbitrage between a DEX and a CEX. The algorithm allows to set up a two-point or triangular arbitrage system. Our system is successfully deployed on two different blockchains. Subsection 4 reports empirical results for the DeFiChain[8] blockchain between November 27, 2023, and December 31, 2023. Two arbitrage systems are deployed on the DeFiChain. The first system (dMFP) contains one Solana arbitrage bot, and the second system (dD6K) combines an Ethereum and a Solana trading bot to efficiently combine exposure liquidity. To test whether our

[8]See DeFiChain (2019).

arbitrage system also works on other blockchains, section 5 reports the successful deployment of the same arbitrage trading algorithm on the SEI[9] blockchain, i.e. third system (0x2f). Section 6 concludes the paper.

2 Mathematical foundations

2.1 Slippage

The volume of a trade interacting with an AMM DEX directly influences the spot price. Consequently, arbitrage traders must assess the anticipated price impact of their orders, quantified as slippage. Arbitrage traders incorporate the expected cost of slippage into their quotation decisions. Xu et al. (2023) define slippage as the difference between the spot price at the initiation of a trade and the actual price realized upon its execution.[10] The following demonstrates how to compute slippage for a DEX order and a market order on a CEX. The DEX AMM conversion function enables the deterministic calculation of expected slippage. Calculating slippage at a DEX is contingent upon the protocol's underlying conversion function. Xu et al. (2023) provide a comprehensive overview of various AMM protocols, detailing each protocol's conversion function, spot exchange rate, slippage, and divergence loss. Depending on the exchange's conversion function, this systematic framework assists investors in identifying embedded risks and predicting expected trade outcomes.

To maintain simplicity, this paper adopts the variables presented by Xu et al. (2023) to demonstrate the conversation function, exchange rate, and slippage calculation at the DEX. The DeFiChain decentralized exchange is a fork of Uniswap V2, and correspondingly, its conversion function is defined as:[11]

$$K = r_1 \cdot r_2 \tag{2.1}$$

where K denotes the invariant, and r_1 and r_2 denote the reserves for asset 1 and asset 2, respectively. At each point in time t, the spot exchange rate, E_2, is given by:

$$E_2 = \frac{r_1}{r_2} \tag{2.2}$$

When a trader interacts with the pool to exchange assets, they provide an amount x_1 to the pool to receive an amount x_2. The expected slippage for this trade, s_{DEX}, is:

$$s_{\text{DEX}} = \frac{\frac{x_1}{x_2}}{\frac{r_1}{r_2}} - 1 = \frac{x_1}{r_1} \tag{2.3}$$

which represents the ratio of the actual price achieved upon trade execution to the spot exchange rate.

The order matching process in a limit order book system operates differently from that of the AMM model, which changes the computation of the expected slippage at CEXs. For limit orders, there is no slippage since they are executed at a fixed price. However, market orders can exhibit slippage costs. Therefore, arbitrage traders determine the expected slippage by fetching current quotes from the limit order book. For a market order, the realized price $P_{\text{realized}}(x)$ is computed as follows:

$$P_{\text{realized}} = \frac{\sum_{i=1}^{n}(x_i \cdot P_i)}{x} \tag{2.4}$$

where x_i and P_i represent the trading volume and price for quote i, respectively, and x denotes the total

[9] See SeiLabs (2024).
[10] See Xu et al. (2023) p. 238
[11] In section 5, we deploy our arbitrage trading system on the Balancer folk. Here, the constant product AMM conversation function includes a weighting of reserves.

trading volume on the CEX, calculated as $\sum_{i=1}^{n} x_i$. Consequently, the slippage at the CEX, s_{CEX}, is determined by:

$$s_{\text{CEX}} = \frac{P_{\text{realized}}}{P_{\text{spot}}} - 1 \tag{2.5}$$

where P_{spot} denotes the spot price at the CEX. Calculating expected slippage in a limit order book exchange is particularly challenging and prone to inaccuracies for two reasons. First, high-frequency traders frequently adjust their quoted prices P_i and volumes x_i, resulting in greater liquidity volatility in the order book compared to the more stable, long-term liquidity provision at DEXs, which change less frequently. Second, platforms such as KuCoin[12] allow orders to be hidden within the order book, making it difficult to assess expected slippage at the CEX accurately.

Many researchers conducting back-testing of arbitrage profits use historical datasets and overlook hidden costs, such as slippage. However, incorporating these costs is essential, as they can significantly affect the accuracy of profit estimates. Neglecting slippage can lead to overestimated arbitrage opportunities. Accounting for slippage and other transaction costs provides a more realistic evaluation of profitability in practical trading environments. Additionally, arbitrage traders often forgo potential profits due to the uncertainty associated with slippage effects. To avoid unprofitable trades, practical arbitrage traders are reluctant to quote break-even profit trading volumes, creating another disparity between the theoretical back-testing results and the practical execution of arbitrage trades.

2.2 Profit and cost function

The slippage and trading costs reduce the realized profit from an arbitrage trade. The costs c_{trade} for a two-point and triangular arbitrage trade include all fees and slippages from the CEX and DEX. The following cost model assumes that a two-point arbitrage trade involves one limit order at the limit order book on the CEX and one order on the AMM DEX. For triangular arbitrage, the model assumes a limit order and a market order on the CEX, mimicking a trading pair on the DEX, along with one trade on the DEX. The trade cost is a variable cost and depends on the volume traded at the DEX and the volume traded at the CEX. For two-point arbitrage, the trade cost $c_{\text{trade}}^{two-point}$ is calculated as:

$$c_{\text{trade}}^{two-point} = f_{\text{CEX}_{\text{limit}}} + f_{\text{DEX}} + s_{\text{DEX}} \tag{2.6}$$

where $f_{\text{CEX}_{\text{limit}}}$ denotes the fee for placing a limit order at the CEX and f_{DEX} denotes the fee for a trade at the DEX. For triangular arbitrage, the trade cost $c_{\text{trade}}^{triangular}$ additionally includes the fee for placing a market order $f_{\text{CEX}_{\text{market}}}$ and the slippage s_{CEX} at the CEX, and is calculated as:

$$c_{\text{trade}}^{triangular} = f_{\text{CEX}_{\text{limit}}} + f_{\text{CEX}_{\text{market}}} + f_{\text{DEX}} + s_{\text{CEX}}{}^{13} + s_{\text{DEX}} \tag{2.7}$$

The profit function π for two-point and triangular arbitrage trades is:

$$\pi = \underbrace{(\rho - c_{\text{trade}} - \Delta p_{\text{DEX}} - \Delta p_{\text{CEX}})}_{\text{realized spread}} \cdot x - F_{\text{transaction}} \tag{2.8}$$

where ρ denotes the configured cross-exchange arbitrage spread, Δp_{CEX} and Δp_{DEX} indicate the price change in bps at the CEX and DEX between quotation and execution, and $F_{\text{transaction}}$ denotes the fixed fee for every transaction on the blockchain. Note that for most CEXs, $f_{\text{CEX}_{\text{limit}}} < f_{\text{CEX}_{\text{market}}}$, since limit orders provide liquidity to the order book, while market orders take liquidity. Exchanges aim to increase

[12] See KCS.Foundation (2022).

[13] This assumes that the trading bot utilizes a market order, which is subject to slippage.

liquidity by incentivizing liquidity providers. At KuCoin,[14] traders exceeding a 30-day trading volume of 350 million USDT do not pay market-making fees, resulting in $f_{\text{CEX}_{\text{limit}}} = 0$. Besides trading costs, an additional cost occurs due to liquidity rotation. C_{travel} represents the cost incurred when circulating liquidity between exchanges and is calculated as follows:

$$C_{\text{travel}} = F_{\text{withdrawal}} + F_{\text{wrapping}} \tag{2.9}$$

where $F_{\text{withdrawal}}$ denotes the withdrawal cost from the CEX, and F_{wrapping} represents the wrapping fees through the wrapping provider. A reduced cost function allows a decrease in ρ while maintaining the same profit level, facilitating earlier execution of the limit order. Consequently, a reduction in ρ leads to a narrowing of the cross-exchange spread between the two exchanges. If the exposure position of the arbitrage portfolio runs low at one exchange, it requires to circulate funds, leading to costs of funds that travel across exchanges C_{travel}. For the majority of CEXs, $F_{\text{withdrawal}}$ is a fixed fee that occurs whenever the funds are withdrawn from the exchange. Additionally, the circulation of funds may demand to use a wrapping service, where the wrapping fee F_{wrapping} adds to travel costs.[15] The results in section 4 show how costly the rotation of liquidity can be. One way to reduce travel costs is the reduction of the desired spread ρ for one order, either bid or ask limit order, when the bot's liquidity runs low at either exchange. The reduced spread level increases the probability of trade executions in one direction and, correspondingly, leads to the autorebalancing effect of the exposure position. The trading system on the SEI blockchain advances our algorithm and incorporates the autorebalancing effect. The flexible spread management decreases C_{travel} but comes at a trade-off for a reduced realized profit or even unprofitable trades. Another option to reduce C_{travel} is to increase the exposure position of the arbitrage portfolio. As $F_{\text{withdrawal}}$ and F_{wrapping} tend to be fixed fees (dependent on arbitrage environment), arbitrage traders can reduce travel costs C_{travel} with an increased exposure portfolio that allows to reduce the frequency of circulations. This allows an increase in the absolute profit but comes at the cost of a lower rate of return due to the increased portfolio position.

The cost functions c_{trade} and C_{travel} are dependent on the exchanges, trading pairs, wrapping providers, and blockchains. A single cost element can be influential for the entire strategy of the algorithm. For example, the Ethereum blockchain has comparable high fees for transactions $F_{\text{transaction}}$that varied between June 3, 2024, and June 3, 2023, between 5$ and 85$. To achieve a profitable cross-exchange arbitrage trade between a CEX and an Ethereum-based DEX, it requires a high limit order volume x at a given spread level ρ to guarantee a profitable trade. Note, if x is low, it might result in unprofitable trade due to the fixed transaction cost $F_{\text{transaction}}$. Another problem can occur if the arbitrage trader quotes a high limit order volume x at a spread ρ that exceeds trading costs, as it can not be guaranteed that the limit order will be completely filled. A fractional fulfillment of the limit order forces the arbitrage trader to trade-off between an unprofitable arbitrage trade and an unbalanced arbitrage portfolio. To circumvent the issue, one strategy is to include a CEX that uses a market order to guarantee a balanced arbitrage trade. However, a market order with a large volume at the CEX comes at the cost of an increased slippage at the CEX s_{CEX}. For blockchains that have a high transaction fee, arbitrage trading between two DEXs could be an alternative, as the expected slippage and order fulfillment can be predetermined more accurately. Concluding, the design of the arbitrage algorithm is heavily dependent on the cost function of the arbitrage environment. Expanding research into arbitrage trading strategies under different cost functions can provide a deeper understanding of practical arbitrage strategies.

[14] See official website.
[15] Depending on the wrapping provider, the fee is fixed or variable based on volume.

2.3 Mempool calculations and transaction sequences

Trading at CEXs heavily depends on execution speed. The execution of trades on the blockchain is not only contingent on execution speed but also on the blockchain consensus mechanism and its corresponding transaction sequence. The following section shows that uncertainty about the spot price at the DEX remains if the transaction sequence of the blockchain is not known. Further, to receive accurate prices at the DEX $P^{\text{DEX}}_{\text{x}_1/\text{x}_2}$, traders should not rely on the price data received from the smart contract. Instead, they should incorporate mempool transactions to determine the live price at the DEX. Transactions that are sent but not yet mined are located in the mempool.[16] Once a new block is found, the smart contract reserves are updated. The time delay and corresponding outdated smart contract reserves impose the risk of a high price deviation Δp_{DEX} at the DEX. The longer the average expected block time, the longer it takes to update the spot price which can lead to a higher risk of a spot price deviation. Every transaction in the mempool requires to be analyzed if it interacts with the trading pair of interest. If a transaction has an influence on the price of the trading pair $P^{\text{DEX}}_{\text{x}_1/\text{x}_2}$, it requires the arbitrage trader to compute the actual price $\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2}$ at the DEX. Recall the conversion function from section 2.1 and assume a mempool containing two raw transactions, a_0 at t_0 and a_1 at t_1. For the first trade a_0, the pool receives the amount x_1, and the trader receives the amount x_2. The new reserve level $\tilde{r}_{1,\text{a}_0}$ in the pool is calculated as:

$$\tilde{r}_{1,\text{a}_0} = r_1 + x_{1,\text{a}_0} \tag{2.10}$$

Then, the new level of reserve $\tilde{r}_{2,\text{a}_0}$ in the pool is determined from the conversation function as:

$$K = \tilde{r}_{1,\text{a}_0} \cdot \tilde{r}_{2,\text{a}_0} \implies \tilde{r}_{2,\text{a}_0} = \frac{K}{\tilde{r}_{1,\text{a}_0}} = \frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}} \tag{2.11}$$

and the trader receives the output $x_2 = \tilde{r}_{2,\text{a}_0} - r_2 = \frac{r_1 \cdot r_2 \cdot (r_1 - x_{1,\text{a}_0})}{r_1 + x_{1,\text{a}_0}}$. The actual price after the trade a_0 is:

$$\hat{P}^{\text{DEX}}_{1/2,\text{a}_0} = \frac{\tilde{r}_{1,\text{a}_0}}{\tilde{r}_{2,\text{a}_0}} = \frac{(r_1 + x_{1,\text{a}_0})}{\frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}}} = \frac{(r_1 + x_{1,\text{a}_0})^2}{r_1 \cdot r_2} \tag{2.12}$$

For the second trade a_1, the pool receives the amount x_2, and the trader receives the amount x_1. The new reserve level $\tilde{r}_{2,\text{a}_1}$ in the pool after the second trade is:

$$\tilde{r}_{2,\text{a}_1} = \tilde{r}_{2,\text{a}_0} + x_{2,\text{a}_1} = \frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}} + x_{2,\text{a}_1} \tag{2.13}$$

with the new level of reserve $\tilde{r}_{(1,a_1)}$ in the pool as:

$$K = \tilde{r}_{1,\text{a}_1} \cdot \tilde{r}_{2,\text{a}_1} \Rightarrow \tilde{r}_{1,\text{a}_1} = \frac{K}{\tilde{r}_{2,\text{a}_1}} = \frac{r_1 \cdot r_2}{\frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}} + x_{(2,a_1)}} \tag{2.14}$$

and the trader receives the output $x_1 = \tilde{r}_{1,\text{a}_1} - \tilde{r}_{1,\text{a}_0} = \frac{r_1 \cdot r_2}{\frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}} + x_{2,\text{a}_1}} - (r_1 + x_{1,\text{a}_0})$. The actual price after the second trade a_1 is:

$$\hat{P}^{\text{DEX}}_{1/2,\text{a}_1} = \frac{\tilde{r}_{1,\text{a}_1}}{\tilde{r}_{2,\text{a}_1}} = \frac{\frac{r_1 \cdot r_2}{\frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}} + x_{2,\text{a}_1}}}{\frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}} + x_{2,\text{a}_1}} = \frac{r_1 \cdot r_2}{\left(\frac{r_1 \cdot r_2}{r_1 + x_{1,\text{a}_0}} + x_{2,\text{a}_1}\right)^2} \tag{2.15}$$

Equations from 2.10 to 2.15 show how to compute the actual price $\hat{P}^{\text{DEX}}_{1/2,\text{a}_1}$ at the DEX before a new block is found based on fair ordering.[17] The arbitrage traders compute the actual price and adjust their

[16] This can vary depending on the blockchain.

[17] The mempool calculation is based on the Uniswap V2. However, to apply this calculation to other DEXs, it must be adjusted to their underlying conversation function.

order to keep their desired spread. Uncertainty remains if the sequence of trade execution is unknown.

Assume the order of these two raw transactions changes, so a_1 happens at t_0 and a_0 at t_1. This is more likely to happen if miners sort transactions based on their transaction fees, i.e., fee ordering, rather than fair ordering (timestamp-based). The order a_1 is sent at t_1 but it is executed first if $F_{\text{transaction}}$ of a_1 is larger than $F_{\text{transaction}}$ of a_0 that is sent at t_0. Then, for the first trade a_1, the pool receives the amount of x_2, and the trader receives the amount of x_1. The new level of reserve $\tilde{r}_{2,\mathrm{a}_1}$ in the pool after the first trade is calculated as follows:

$$\tilde{r}_{2,\mathrm{a}_1} = r_2 + x_{2,\mathrm{a}_1} \tag{2.16}$$

Then, the new level of reserve $\tilde{r}_{1,\mathrm{a}_1}$ in the pool is:

$$\tilde{r}_{1,\mathrm{a}_1} = \frac{K}{\tilde{r}_{2,\mathrm{a}_1}} = \frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}} \tag{2.17}$$

and the trader receives the output $x_1 = \tilde{r}_{1,\mathrm{a}_1} - r_1 = \frac{r_1 \cdot r_2 \cdot (r_2 - x_{2,\mathrm{a}_1})}{r_2 + x_{2,\mathrm{a}_1}}$. The actual price after the first trade a_1 is:

$$\hat{P}^{\text{DEX}}_{1/2,\mathrm{a}_1} = \frac{\tilde{r}_{1,\mathrm{a}_1}}{\tilde{r}_{2,\mathrm{a}_1}} = \frac{\frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}}}{r_2 + x_{2,\mathrm{a}_1}} = \frac{r_1 \cdot r_2}{(r_2 + x_{2,\mathrm{a}_1})^2} \tag{2.18}$$

For the second trade a_0, the pool receives the amount of x_1, and the trader receives the amount of x_2. The new level of reserve $\tilde{r}_{1,\mathrm{a}_0}$ in the pool is:

$$\tilde{r}_{1,\mathrm{a}_0} = \tilde{r}_{1,\mathrm{a}_1} + x_{1,\mathrm{a}_0} = \frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}} + x_{1,\mathrm{a}_0} \tag{2.19}$$

with the new level of reserve $\tilde{r}_{2,\mathrm{a}_0}$ in the pool after the second trade as:

$$\tilde{r}_{2,\mathrm{a}_0} = \frac{K}{\tilde{r}_{1,\mathrm{a}_0}} = \frac{r_1 \cdot r_2}{\frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}} + x_{1,\mathrm{a}_0}} \tag{2.20}$$

and the trader receives the output $x_2 = \tilde{r}_{2,\mathrm{a}_0} - \tilde{r}_{2,\mathrm{a}_1} = \frac{r_1 \cdot r_2}{\frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}} + x_{1,\mathrm{a}_0}} - (r_2 + x_{2,\mathrm{a}_1})$. The actual price after the second trade a_0 at $t = 1$ is:

$$\hat{P}^{\text{DEX}}_{1/2,\mathrm{a}_0} = \frac{\tilde{r}_{1,\mathrm{a}_0}}{\tilde{r}_{2,\mathrm{a}_0}} = \frac{\frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}} + x_{1,\mathrm{a}_0}}{\frac{r_1 \cdot r_2}{\frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}} + x_{1,\mathrm{a}_0}}} = \frac{\left(\frac{r_1 \cdot r_2}{r_2 + x_{2,\mathrm{a}_1}} + x_{1,\mathrm{a}_0}\right)^2}{r_1 \cdot r_2} \tag{2.21}$$

Equations from 2.16 to 2.21 show how to compute the actual price $\hat{P}^{\text{DEX}}_{1/2,\mathrm{a}_1}$ at the DEX before a new block is found based on fee ordering. Table 2.1 highlights the nuanced differences in spot prices based on the order sequence of the two trades, emphasizing how structural differences in the formulas affect price sensitivity and stability. In fair ordering, the numerator of the spot price represents the product of the initial reserves, which is a constant, while the denominator depends on the levels of x_{1,a_0} and x_{2,a_1}. This implies that any increase in x_{2,a_1} will decrease the denominator, thereby reducing the spot price. Moreover, the squared denominator amplifies this effect, making the spot price more sensitive to adjustments.

In contrast, under fee ordering, the numerator of the spot price depends on the levels of x_{2,a_1} and x_{1,a_0}; thus, any increase in x_{1,a_0} raises the numerator, increasing the spot price. Note that any increase in transaction fees can change the order sequence in fee ordering. For fair ordering, a change in order sequence is less likely. However, Baum et al. (2022) argue that even under fair ordering, front-running attacks remain a threat, as an attacker can rush transactions within the network.[18] Consequently, traders

[18] See Baum et al. (2022) p. 5.

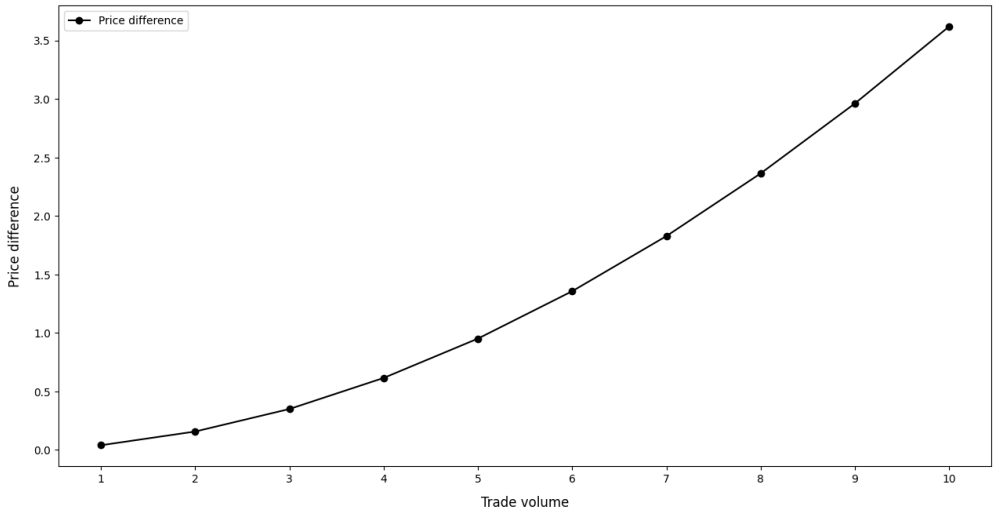
Table 2.1: Comparison of Fair and Fee Ordering Effects on Spot Prices

	First Trade	Second Trade
Fair Ordering:		
$a_0, t_0 \Rightarrow a_1, t_1$	$\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_0} = \frac{(r_1+x_{1,\text{a}_0})^2}{r_1 \cdot r_2}$	$\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_1} = \frac{r_1 \cdot r_2}{\left(\frac{r_1 \cdot r_2}{r_1+x_{1,\text{a}_0}}+x_{2,\text{a}_1}\right)^2}$
Fee Ordering:		
$a_1, t_1 \Rightarrow a_0, t_0$	$\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_1} = \frac{r_1 \cdot r_2}{(r_2+x_{2,\text{a}_1})^2}$	$\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_0} = \frac{\left(\frac{r_1 \cdot r_2}{r_2+x_{2,\text{a}_1}}+x_{1,\text{a}_0}\right)^2}{r_1 \cdot r_2}$

can never be fully certain of the final transaction sequence under either fair or fee ordering.

Figure 2.1 depicts the simulation of spot price deviation conditional on trade sequences. The following simulation assumes a trade volume for a_0 as $\frac{x_1}{r_1}$ and for a_1 as $\frac{x_2}{r_2}$. The trades are executed in the following order $[a_0, a_1]$ and $[a_1, a_0]$. The initial amount for reserve r_1 and reserve r_2 are 750,000 USDT and 10 BTC. We calculate the spot price after $[a_0, a_1]$ as $\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_1}$ and after $[a_1, a_0]$ as $\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_0}$, and the spot price change as $\Delta\hat{P}^{\text{DEX}} = \frac{\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_1}}{\hat{P}^{\text{DEX}}_{\text{x}_1/\text{x}_2,\text{a}_0}}$. We repeat these calculations for varying trading volume where $\frac{x_1}{r_1} = \frac{x_2}{r_2}$ at t_0 takes values in the interval $[0.001, 0.1]$, increasing the trading volume by 1% in each step.[19]

Figure 2.1: Spot price change



The y-axis depicts the spot price change in basis points for different volume levels. The results show that for a small trading volume, the order sequence has a minor influence. However, with rising trading volume, the relevance of the order sequence gains importance. For example, with highly volatile assets traded using a constant product AMM, the transaction sequence can result in significant spot price differences. Miners can exploit the transaction sequence to enhance their exposure position without engaging in any trading activity.

[19]The same simulation can be replicated with different levels of reserves and coins that lead to identical results.

To sum up, successful high-frequency trading at the CEX is primarily dependent on a good network connection. In contrast, trading at the DEX also depends on the blockchain's consensus mechanism, which determines the transaction sequence. This sequence can lead to front-running attacks, potentially resulting in unprofitable trades. Baum et al. (2022) propose fair ordering as a transaction sequencing method to reduce front-running attacks; however, as they state, even fair ordering cannot completely mitigate front-running. An attacker who identifies a trade in the mempool that significantly moves the DEX price by $s_{\text{DEX}}(x)$ can front-run the trader with a trade x_1 by paying an increased transaction fee $F_{\text{transaction}}(x_1)$. Additionally, the attacker can "sandwich" the trader by closing their position with a second trade x_2 to recover the initial asset x_1.

If the attacker increases their exposure position Δx_1 and the monetary value of $\Delta x_1 > F_{\text{transaction}}(x_1) + F_{\text{transaction}}(x_2)$, the sandwich attack is successful. To avoid becoming a victim of sandwich attacks, arbitrage traders can choose a trading volume such that $s_{\text{DEX}}(x)$ remains low, which reduces economic opportunities for attackers. Daian et al. (2019) describe how front-running and price gas auctions between multiple attackers have led to increased transaction fees on the Ethereum network. Solana recently introduced priority fees, which might pave the way for price gas auctions on Solana and lead to increased transaction costs in the future.[20]

3 Arbitrage trading bot

3.1 Setup

To set up an efficient arbitrage bot, a developer requires a good connection to the network. Garvey and Wu (2010) conduct an analysis of 2,000 stock traders and find that being geographically closer to the exchange enhances execution speed and reduces execution costs. Additionally, they observe that traders situated nearer to the exchange's computers are more likely to engage in time-sensitive strategies. Unlike CEXs, DEXs run through blockchain validators that are geographically dispersed. Therefore, it is challenging to optimize the location to reduce execution speed. However, the majority of blockchains transparently show the geographical distribution of blockchain validators. Therefore, when configuring a server-hosted blockchain node, the server's location should be strategically positioned near the current blockchain validators to reduce latency effectively. Beside the reduced latency and continuous network connection, a server-hosted blockchain node provides additional advantages over a local node. The server infrastructure offers a high level of security, significantly reducing the risk of cyber-attacks. For example, the Bitvise SSH client management system provides a secure means of connecting a local computer to a remote server.[21]

One key feature is SSH key encryption, which ensures a secure method of server authentication. Users generate an SSH key pair consisting of a public and private key. The public key is stored on the server, while the private key remains on the local computer. During any login attempt, the local SSH client encrypts messages using the private key, which the server decrypts with its public key, enabling secure data transfer. However, the local private key poses a security risk. If someone gains access to the local device, they could authenticate to the server. To mitigate this risk, the private key can be encrypted with a password, adding an additional layer of security.

Further, a server-hosted node offers a distinct economic advantage by eliminating the initial costs associated with purchasing hardware for setting up a node. It provides a flexible solution to scalability challenges. As blockchain software evolves, server-based solutions are advantageous in ensuring the infrastructure can adapt to growing resource demands. Summing up, a server infrastructure enhances the uptime, security, flexibility, and efficiency of the arbitrage system.

[20] See official website.
[21] To our knowledge, this client can only be used with Microsoft Windows operating systems.

In addition to these economic and scalability benefits, maintaining a stable and up-to-date connection with the blockchain network is crucial for the arbitrage system's reliability. Since the DEX operates as a smart contract on the blockchain, the node automatically synchronizes with the network to retrieve the latest on-chain data. However, network disruptions or corrupt data can cause the node to lose connection to the blockchain network, leading to outdated price information from the DEX and preventing the execution of trades. In such cases, the trading bot should pause operations until the node is reconnected to the network. Restarting the node can often resolve the issue. In more severe situations, the current block data may need to be deleted, requiring a download of a network snapshot that contains the most recent data. It is crucial to obtain this snapshot from an official source to prevent the risk of malicious malware.

In contrast to the node that is hosted on the server to interact with the DEX, KuCoin operates as a centralized exchange, which is privately managed and acts as a custodian for traders. Data is provided through RESTful API and WebSocket protocols, enabling communication between the server and client. While the RESTful API requires individual HTTP requests for each data retrieval, a WebSocket facilitates a continuous data transfer, making it the preferred option for high-frequency trading due to its faster updates. To authenticate the server with the client, users must generate an API key, API secret, and passphrase. To further enhance security, KuCoin allows the configuration of IPv6 addresses to restrict data access to specified addresses. In this setup, we configure the server's IPv6 address as the only permitted address for API connections, adding an extra layer of security by preventing unauthorized withdrawals, even if API credentials are compromised.

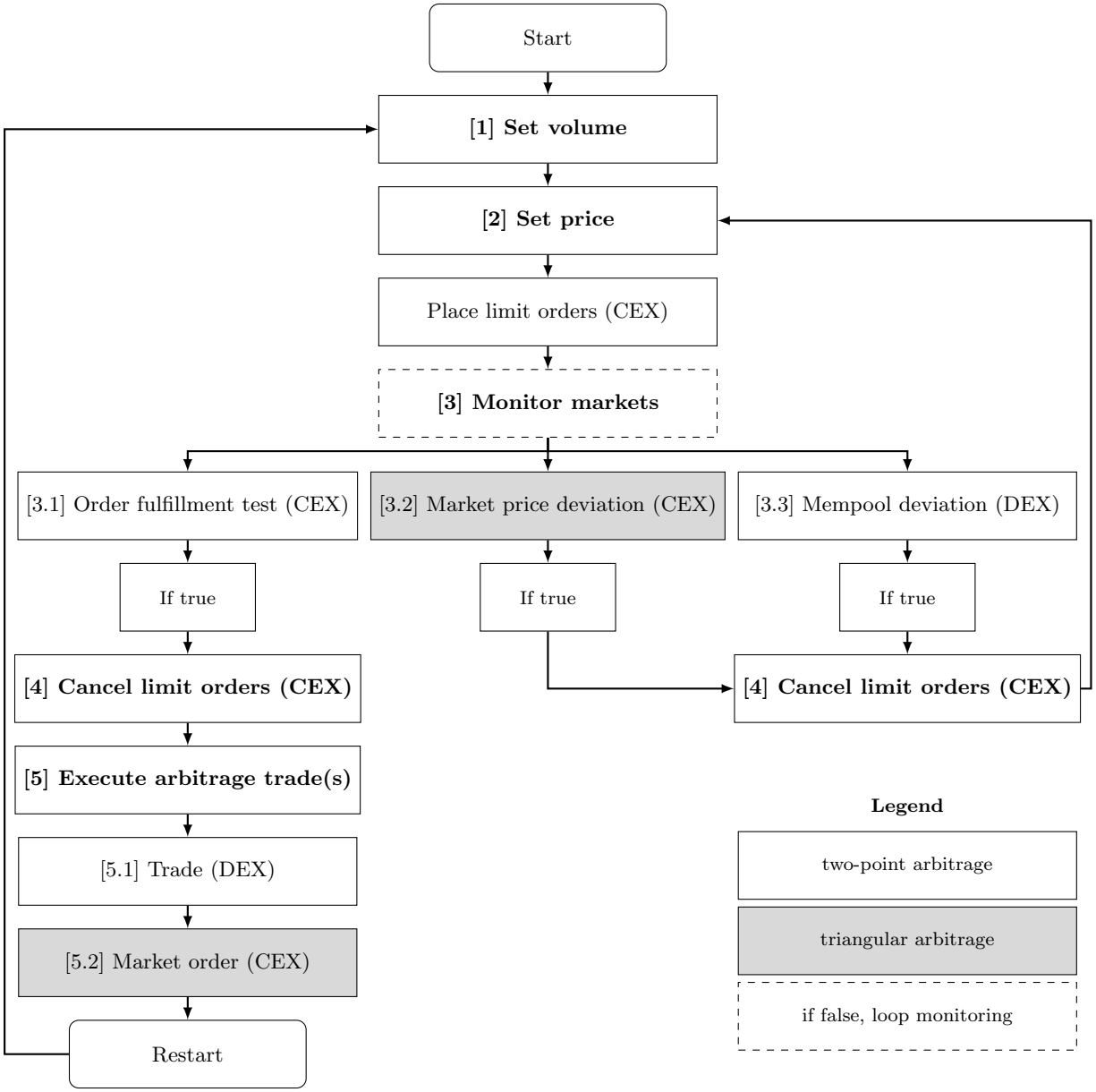
Additionally, we use Crontab, which enhances the arbitrage system in several ways. First, for reporting purposes, a separate Python script is deployed on the server to document the trading bot's overall exposure position, with updates sent to the private Telegram group every minute. Second, KuCoin reduces trading fees by 20% for users holding KuCoin tokens (KCS). When the KCS function is enabled, trading fees are deducted from the KCS balance. Another Python script is scheduled to monitor the KCS balance and automatically purchase additional KCS if it falls below a preset threshold. This ensures a minimized KCS exposure position while maintaining the 20% fee reduction. Third, for every triangular arbitrage trade, the system tracks the DEX order ID and the limit or market order IDs from the CEX. A further Python script retrieves trade data from each exchange using these IDs to calculate the realized profit.

3.2 Algorithm

In this subsection, we outline the order placement and execution strategy for two-point and triangular arbitrage between a decentralized and centralized cryptocurrency exchange. We outline the design of the trading bots, detailing the conditions for canceling a quote, managing limit order fulfillment, and how the bot's algorithm can minimize slippage costs. Additionally, we discuss the algorithmic distinctions between two-point and triangular arbitrage strategies, along with the calculations performed at each step. In section 4, we present the practical outcomes of the trading bots. The arbitrage system on the SEI blockchain executes two-point arbitrage, while the system on the DeFiChain employs triangular arbitrage. In the following, the algorithms are explained in relation to the corresponding trading pairs at the DeFiChain but are adaptable to other trading pairs across cryptocurrency markets.

Figure 3.1 depicts the design of the arbitrage bot in six sequential steps, where the two-point arbitrage strategy operations are marked in white boxes, and the triangular arbitrage bot's additional computational tasks are marked in grey boxes. The reason for the triangular arbitrage strategy is the unavailability of equal trading pairs at both exchanges. For our trading bot, we aim to arbitrage the DFI-SOL pool at the DEX. However, the DEX is the only exchange that lists that specific trading pair.

Figure 3.1: Bot desing



Therefore, we mimic the DFI-SOL trading pair with two trading pairs at the CEX: DFI-USDT and SOL-USDT. Any bot design that necessitates executing a market order within a limit order book system is inherently risky due to unpredictable slippage costs. If the liquidity of a market is low, the slippage costs increase. As a preparatory step, it is essential to assess the liquidity depth in the DFI-USDT and SOL-USDT markets on the central exchange. Brauneis et al. (2021) describe and compare high-frequency, easy-to-compute liquidity measures to develop liquidity proxies. A separate Python script retrieves order book data on a minute-by-minute basis to calculate the expected slippage for various trade volumes, as described in section 2. Unlike the SOL-USDT market, which has sufficient liquidity depth, we identify that the DFI-USDT market exhibits low liquidity, which makes it costly to work with a market order.

In the following, we describe the bot design, referring to figure 3.1 in every detail. To avoid slippage costs, the bot algorithm starts **[1]** with two limit orders in the DFI-USDT market at the CEX $P^{\text{CEX}}_{\text{DFI}-\text{USDT}}$ that wait to be executed. A bid $P^{\text{CEX}^{\text{bid}}}_{\text{DFI}-\text{USDT}}$ and ask $P^{\text{CEX}^{\text{ask}}}_{\text{DFI}-\text{USDT}}$ limit order is placed around the spot price to enable a limit order fulfillment if the market moves in either direction. Placing two limit orders requires determining the appropriate limit order prices $P^{\text{CEX}^{\text{ask}}}_{\text{DFI}-\text{USDT}}$ and $P^{\text{CEX}^{\text{bid}}}_{\text{DFI}-\text{USDT}}$, and bid and ask volumes, bid_{vol} and ask_{vol} respectively. The exposure positions on both exchanges decide upon the trade volume size. For the bid order, the trader aims to buy DFI liquidity $L^{\text{CEX}}_{\text{DFI}}$ at the CEX with USDT funds $L^{\text{CEX}}_{\text{USDT}}$. Balancing the two-point arbitrage trade requires an equivalent USDT value in DFI $L^{\text{DEX}}_{\text{DFI}} \cdot P^{\text{CEX}}_{\text{DFI}-\text{USDT}}$ to be sold at the DEX. Therefore, the maximum possible trade volume is determined by the minimum value of different exposure positions and a maximum volume cap designed to limit slippage costs. The volumes for two-point arbitrage are calculated as follows:

$$\text{bid}_{\text{vol}} = \min(L^{\text{CEX}}_{\text{USDT}}, L^{\text{DEX}}_{\text{DFI}} \cdot P^{\text{CEX}}_{\text{DFI}-\text{USDT}}, \text{max}_{\text{vol}}) \tag{3.1}$$

$$\text{ask}_{\text{vol}} = \min(L^{\text{DEX}}_{\text{USDT}}, L^{\text{CEX}}_{\text{DFI}} \cdot P^{\text{CEX}}_{\text{DFI}-\text{USDT}}, \text{max}_{\text{vol}}) \tag{3.2}$$

The volumes for triangular arbitrage are calculated as follows:

$$\text{bid}_{\text{vol}} = \min(L^{\text{CEX}}_{\text{USDT}}, L^{\text{DEX}}_{\text{DFI}} \cdot P^{\text{CEX}}_{\text{DFI}-\text{USDT}}, L^{\text{CEX}}_{\text{SOL}} \cdot P^{\text{CEX}}_{\text{SOL}-\text{USDT}}, \text{max}_{\text{vol}}) \tag{3.3}$$

$$\text{ask}_{\text{vol}} = \min(L^{\text{CEX}}_{\text{USDT}}, L^{\text{CEX}}_{\text{DFI}} \cdot P^{\text{CEX}}_{\text{DFI}-\text{USDT}}, L^{\text{DEX}}_{\text{SOL}} \cdot P^{\text{CEX}}_{\text{SOL}-\text{USDT}}, \text{max}_{\text{vol}}) \tag{3.4}$$

If bid_{vol} or $\text{ask}_{\text{vol}} < \text{min}_{\text{vol}}$, then no order is placed. For the triangular bid arbitrage trade, an additional constraint is the SOL position at the CEX $L^{\text{CEX}}_{\text{SOL}} \cdot P^{\text{CEX}}_{\text{SOL}-\text{USDT}}$. In the triangular bid trade, at the CEX, SOL $L^{\text{CEX}}_{\text{SOL}}$ is sold to receive USDT $L^{\text{CEX}}_{\text{USDT}}$ and, then, DFI liquidity $L^{\text{CEX}}_{\text{DFI}}$ is purchased with USDT $L^{\text{CEX}}_{\text{USDT}}$. Meanwhile, on the DEX, DFI $L^{\text{DEX}}_{\text{DFI}}$ is directly sold to purchase SOL $L^{\text{DEX}}_{\text{SOL}}$. Despite the available liquidity at either exchange, the maximum and minimum volume (max_{vol} and min_{vol}) for either limit order cannot be infinitely large or close to zero. Recall from subsection 2.2, an increased trading volume increases the slippage cost and the risk of being sandwiched at the DEX. Therefore, maximum volume controls for excessive slippage costs. KuCoin defines a minimum trade volume for their markets. Thus, minimum volume ensures that no order is placed if the calculated maximum trading volume falls below the minimum trade volume defined by the exchange.

The second parameter for placing a limit order is calculating the limit order prices **[2]**. For two-point arbitrage, to compute the limit order price, the bot retrieves the price for DFI $P^{\text{DEX}}_{\text{DFI}-\text{USDT}}$ at the DEX and adds a configurable spread level τ. For triangular arbitrage, it requires the price of Solana from the CEX $P^{\text{CEX}}_{\text{SOL}-\text{USDT}}$ to express the DFI $P^{\text{DEX}}_{\text{DFI}-\text{SOL}}$ in USDT. The prices for two-point arbitrage are

calculated as follows:

$$P^{\text{CEX}^{\text{bid}}}_{\text{DFI}-\text{USDT}} = P^{\text{DEX}}_{\text{DFI}-\text{USDT}} \cdot (1-\tau) \tag{3.5}$$

$$P^{\text{CEX}^{\text{ask}}}_{\text{DFI}-\text{USDT}} = P^{\text{DEX}}_{\text{DFI}-\text{USDT}} \cdot (1+\tau) \tag{3.6}$$

The prices for triangular arbitrage are calculated as follows:

$$P^{\text{CEX}^{\text{bid}}}_{\text{DFI}-\text{USDT}} = P^{\text{DEX}}_{\text{DFI}-\text{SOL}} \cdot P^{\text{CEX}}_{\text{SOL}-\text{USDT}} \cdot (1-\tau) \tag{3.7}$$

$$P^{\text{CEX}^{\text{ask}}}_{\text{DFI}-\text{USDT}} = P^{\text{DEX}}_{\text{DFI}-\text{SOL}} \cdot P^{\text{CEX}}_{\text{SOL}-\text{USDT}} \cdot (1+\tau) \tag{3.8}$$

Given a specified price and volume, ask and bid limit orders can be placed in the limit order book.

The bot starts the price monitoring process [**3**]. For two-point arbitrage, two parallel processes start with testing for a limit order fulfillment [**3.1**] and price deviations at the DEX that arise through orders in the mempool [**3.3**]. Any significant price change at the DEX must be handled to maintain the demanded spread level. To avoid the time delay until a new block is found, the bot predetermines the current price at the DEX. Refer to section 2.3, which explains how to calculate the effect of trades on the spot price with the help of the conversation function. The bot monitors the mempool and tests for interactions with the DFI-USDT trading pool. For triangular arbitrage, three processes start [**3.1**], [**3.2**], and [**3.3**]. The triangular arbitrage bot requires an additional test [**3.2**] for the price change in SOL $P^{\text{CEX}}_{\text{SOL}-\text{USDT}}$ at the CEX. This is because the DFI price at the DEX depends on price changes of SOL.

For the bot configuration, we define the price deviation threshold of 5 bps. An increased threshold results in fewer order cancellations and extends the duration that orders remain active within the order book system. However, the trade-off is that the realized profit may exhibit greater variability due to the increased tolerance for price deviations, which can lead to less profitable outcomes. The prices reset for two-point arbitrage occurs when:

$$\text{abs}(\Delta P^{\text{DEX}}_{\text{DFI}-\text{SOL}}) > 5\text{bps} \tag{3.9}$$

The prices reset for triangular arbitrage occurs when:

$$\text{abs}(\Delta P^{\text{DEX}}_{\text{DFI}-\text{SOL}}) > 5\text{bps} \quad \text{or} \quad \text{abs}(\Delta P^{\text{CEX}}_{\text{SOL}-\text{USDT}}) > 5\text{bps} \tag{3.10}$$

If either condition [**3.2**] or [**3.3**] is met, old limit orders are canceled, and [**4**] new limit orders are placed to ensure the initial spread level is maintained. The bot can continue using the same volume data, as there has been no change in the exposure position. The bot exits the looped processes if the limit order is filled [**3.1**] and the trade is rebalanced. Frequently, only a fraction of the limit order is filled. If the rest of the limit order is not canceled, either limit order might have additional fillings during the trade handling. To avoid a second filling and guarantee a balanced exposure position, both limit orders are canceled [**4**], and the filled limit order is handled. Please note that at the CEX, the trade volume for the SOL-USDT and DFI-USDT markets is measured in terms of SOL and DFI, respectively. In contrast, the DEX specifies the trade volume based on the input currency.

The bot executes arbitrage trade [**5**] with trade volumes for bid and ask. For two-point trade, bid and ask fulfillment [**5.1**] are:

$$\text{DFI}^{\text{DEX}}_{\text{sell}_{\text{vol}}} = \text{DFI}^{\text{CEX}}_{\text{bid}_{\text{fulfillment}}} \cdot (1+f_{\text{DEX}}) \tag{3.11}$$

$$\text{USDT}^{\text{DEX}}_{\text{sell}_{\text{vol}}} = \text{DFI}^{\text{CEX}}_{\text{ask}_{\text{fulfillment}}} \cdot P^{\text{CEX}}_{\text{DFI}-\text{USDT}} \cdot (1+f_{\text{DEX}}) \tag{3.12}$$

For triangular trade, bid and ask fulfillment **[5.1]** and **[5.2]** are:

$$\mathrm{DFI}^{\mathrm{DEX}}_{\mathrm{sell_{vol}}} = \mathrm{DFI}^{\mathrm{CEX}}_{\mathrm{bid_{fulfillment}}} \cdot (1 + f_{\mathrm{DEX}}) \tag{3.13}$$

$$\mathrm{SOL}^{\mathrm{CEX_{market}}}_{\mathrm{sell_{vol}}} = \mathrm{DFI}^{\mathrm{CEX}}_{\mathrm{bid_{fulfillment}}} \cdot \frac{P^{\mathrm{CEX}}_{\mathrm{DFI-USDT}}}{P^{\mathrm{CEX}}_{\mathrm{SOL-USDT}}} \tag{3.14}$$

$$\mathrm{SOL}^{\mathrm{DEX}}_{\mathrm{sell_{vol}}} = \mathrm{DFI}^{\mathrm{CEX}}_{\mathrm{ask_{fulfillment}}} \cdot \frac{P^{\mathrm{CEX}}_{\mathrm{DFI-USDT}}}{P^{\mathrm{CEX}}_{\mathrm{SOL-USDT}}} \cdot (1 + f_{\mathrm{DEX}}) \tag{3.15}$$

$$\mathrm{SOL}^{\mathrm{CEX_{market}}}_{\mathrm{buy_{vol}}} = \mathrm{DFI}^{\mathrm{CEX}}_{\mathrm{ask_{fulfillment}}} \cdot \frac{P^{\mathrm{CEX}}_{\mathrm{DFI-USDT}}}{P^{\mathrm{CEX}}_{\mathrm{SOL-USDT}}} \tag{3.16}$$

For two-point arbitrage, bid limit order fulfillment in DFI is sold at the DEX $\mathrm{DFI}^{\mathrm{DEX}}_{\mathrm{sell_{vol}}}$ and ask limit order fulfillment in DFI is bought with USDT at the DEX $\mathrm{USDT}^{\mathrm{DEX}}_{\mathrm{sell_{vol}}}$. For triangular bid limit order filling, $\mathrm{SOL}^{\mathrm{CEX_{market}}}_{\mathrm{sell_{vol}}}$ is sold with a market order to refill the $\mathrm{USDT}^{\mathrm{CEX}}$ liquidity that was spent for the bid limit order. Additionally, $\mathrm{DFI}^{\mathrm{DEX}}_{\mathrm{sell_{vol}}}$ is sold to refill the sold $\mathrm{SOL}^{\mathrm{DEX}}$ amount. After ask limit order filling for a triangular arbitrage trade, $\mathrm{USDT}^{\mathrm{CEX}}$ is used to purchase $\mathrm{SOL}^{\mathrm{CEX_{market}}}_{\mathrm{buy_{vol}}}$ liquidity with a market order. Additionally, $\mathrm{SOL}^{\mathrm{DEX}}_{\mathrm{sell_{vol}}}$ is sold to refill the sold $\mathrm{DFI}^{\mathrm{DEX}}$ amount. For the triangular arbitrage strategy, the maximum trading volume requires limits for two reasons. First, as with two-point arbitrage, the slippage at the DEX should be limited. Second, to avoid high slippage costs for the market order at the CEX. Note that the term $(1 + f_{\mathrm{DEX}})$ helps to perfectly balance the volume of the arbitrage trade. The DEX extracts the trading fee from the trade output, unlike the CEX where the trading fee is deducted from the KCS balance. Subsequent to the trading activity, the balances and prices at each exchange vary, demanding a recalculation of the quote variables. Given its new trading volume, the bot starts the initial quotation and monitoring processes. Similarly, as with transactions in **[3.3]**, the bot automatically incorporates its own mempool transaction, adjusts the spot price, and operates with the updated DEX price accordingly. In case the bot runs out of liquidity for either coin at either exchange, a separate script manages the distribution of liquidity. The script is outsourced to avoid additional computations for the trading bot and increase the speed of the bot.

Figure 3.2: Flow between DEX, wrapping provider, and CEX

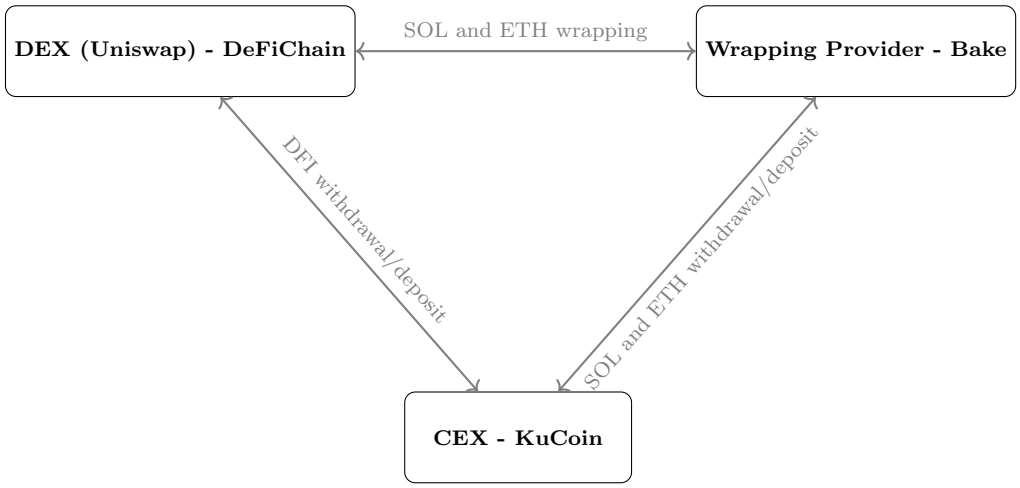


Figure 3.2 depicts the liquidity flow for DFI, SOL, and wrapped SOL (wSOL) on the DeFiChain. Caldarelli (2021) explains that wrapped tokens help to overcome the absence of communication between blockchains. The DeFiChain has no direct connection with the Solana blockchain, which requires a wrapping provider to trade assets from other blockchains. If SOL liquidity runs low at the DEX, the funds are circulated through the wrapping provider (native SOL transaction sends the liquidity from

KuCoin to Bake). Bake Ltd serves as a custodian for native crypto assets on the DeFiChain. It locks the SOL on the Solana blockchain and mints an equivalent amount of wSOL on the DeFiChain. The wSOL tokens are sent via a native DeFiChain transaction from Bake to the bot address to increase wSOL liquidity at the DEX. The approach to use a centralized company as a custodian for all crypto assets bears a high risk for the blockchain and questions the decentralization of the blockchain. To circulate the native DFI coin, a direct withdrawal from Kucoin to DeFiChain and vice versa helps to keep a balanced distribution of DFI funds.

4 Empirical analysis

We deploy the arbitrage system on two different wallet addresses for the time horizon from November 27, 2023, until December 31, 2023.[22] The fee rate for the limit order amounts to $f_{\text{CEX}_{\text{limit}}} = 16$ bps and for the market order $f_{\text{CEX}_{\text{market}}} = 8$ bps. The fee rate at the DEX is $f_{\text{DEX}} = 20$ bps [23]. The spread level is configured at $\rho = 120$ bps. The first triangular arbitrage system, dMFP, executes arbitrage trades for the following trading pairs: DFI-USDT, SOL-USDT, and DFI-SOL. The second triangular arbitrage system, dD6K, runs two parallel trading bots. The first bot operates on the DFI-USDT, SOL-USDT, and DFI-SOL pairs, while the second bot executes arbitrage trades for the trading pairs DFI-USDT, ETH-USDT, and DFI-ETH.

The initial exposure portfolio for the dMFP is 25,000 DFI, 2,100 USDT, and 225 SOL. The maximum trade volume for each transaction is capped at \$1,500 to reduce slippage costs at both exchanges, i.e., $s_{\text{CEX}} + s_{\text{DEX}}$. Based upon the calculations described in section 2.1, the expected slippage for a trade volume of \$1,500 at the CEX, $\text{E}(s_{\text{CEX}})$, ranged from 0 to 2 basis points (bps) and at the DEX, $\text{E}(s_{\text{DEX}})$, ranged from 4 to 6 bps. The variation in slippage at the CEX is attributed to the fluctuating liquidity available, which changes frequently. In contrast, the variation at the DEX is primarily influenced by the total value locked (TVL) in the corresponding trading pair. Although liquidity deposits remain mostly stable, the TVL fluctuates with changes in the price of underlying assets. At the start and end of the trading period, the coin prices on November 27 were: SOL \$59.18, DFI \$0.246, ETH \$2064.07, USDT \$1, and on December 31: SOL \$101.99, DFI \$0.159, ETH \$2294.34, USDT \$1.

The stablecoin provided by Tether Foundation, i.e., USDT, shows little price variation during the time horizon. Caldarelli (2021) explains that stablecoins are designed to maintain their value relative to fiat currency through mechanisms of token minting and burning.[24] These operations are supported by reserves held by trusted entities, which should transparently show their proof of funds. Griffin and Shams (2020) analyze the relationship between USDT and Bitcoin, highlighting risks associated with stablecoin issuers, such as the potential to create tokens without corresponding demand, which can artificially inflate the market. Therefore, Caldarelli (2021) outlines the advantage of non-custodial stablecoins over custodial stablecoins. Non-custodial stablecoins, e.g., USD-J or DAI, are issued by collateralizing assets into a smart contract with a minimum collateral ratio of 150%. This trustless system operates entirely on-chain, allowing investors to verify the stablecoin's value based on the assets locked in the smart contract, providing greater transparency and security.

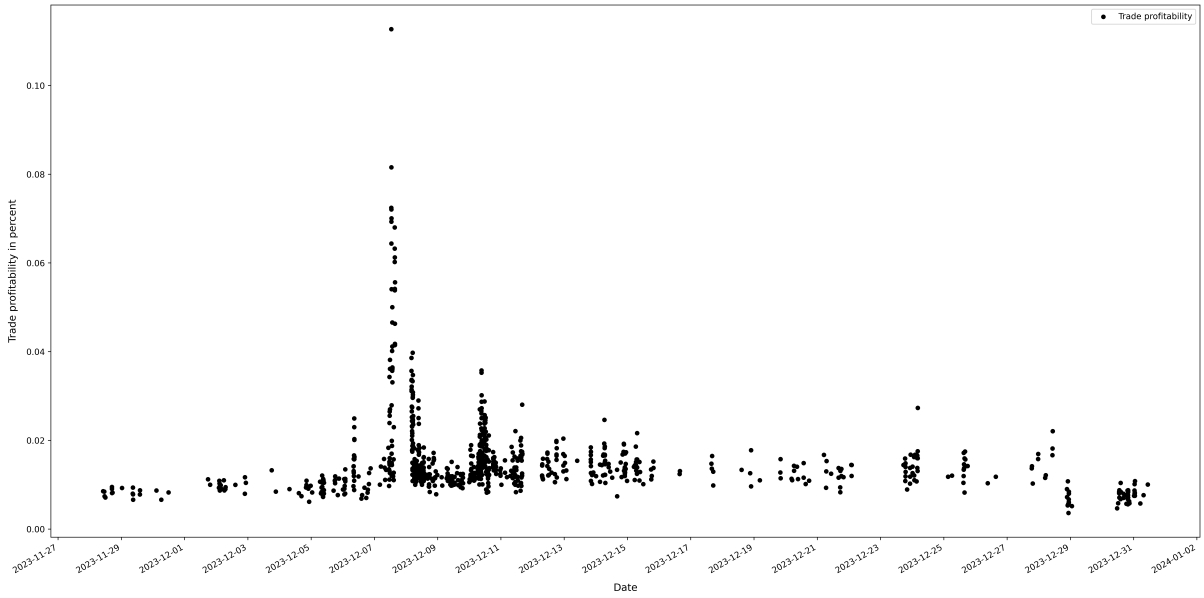
Figure 4.1 illustrates the distribution of realized arbitrage profit in percent for the dMFP. The realized profit incorporates the trading costs C but excludes the cost C_{travel} that occurs whenever liquidity rotates between the exchanges. Each data point represents the realized profit for a triangular arbitrage trade π in percent. The dMFP targets an expected profit of 70 bps. Figure 4.1 shows outliers up to 1070 bps.

[22] The wallet address for dMFP system is dMa2PMnAudfZds8mnRm8EtjDWf7UWAuQFP, and the wallet address for the dD6K system is dDXmMofYrtdiqB4MuGewDEA6jJN8Yo4F6K. More information can be found on the DeFi Scan (see official website).

[23] Trading fees at the CEX are reduced from 20bps to 16 bps for the limit order and from 10 bps to 8 bps for the market order, due to a 20 percent discount applied for holding KCS.

[24] See Caldarelli (2021) p.5.

Figure 4.1: Trade profitability dMFP



In moments of high volatility, the quoted bid/ask order price is higher/lower than the best ask/bid price in the order book, and two limit orders automatically converge at the best ask/bid price, which allows for profits above 70 bps.

Figure 4.2: Cumulative profit dMFP

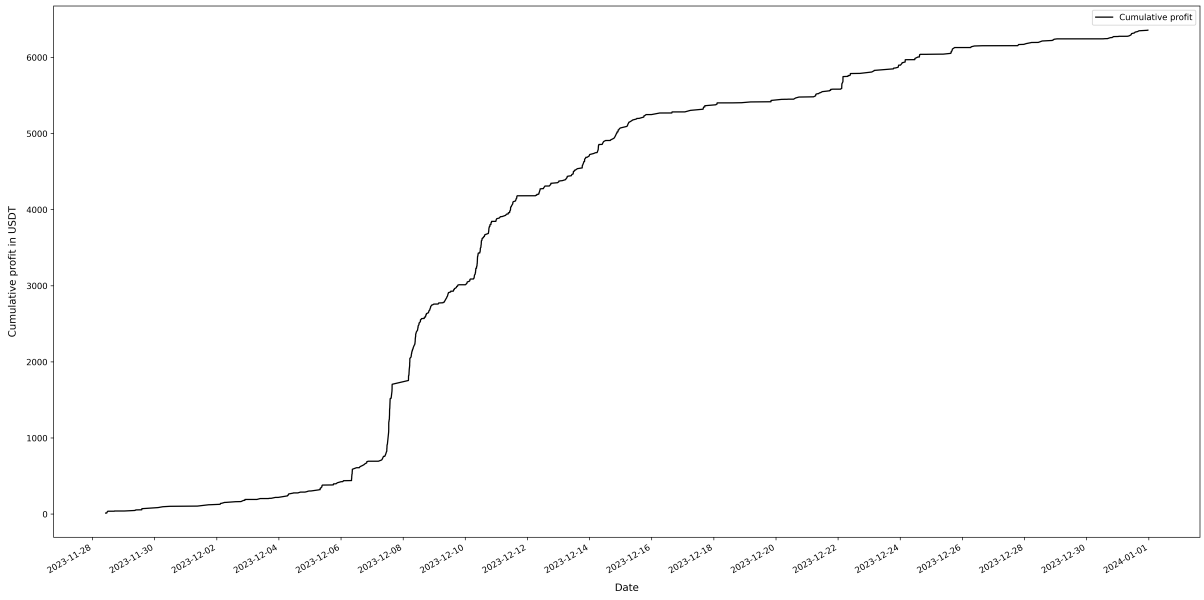


Figure 4.2 shows the cumulative profit of the dMFP over time and depicts a continuous pattern of realized arbitrage opportunities. The total amount of profits is 6357.75 USDT. The most profitable trade yields a profit of 1070 bps, amounting to $160.63 with a single arbitrage trade. For the dMFP, out of 1074 arbitrage trades, not a single trade results in a loss, which indicates a low execution risk. The average realized spread is 154 bps and $5.88 profit per transaction.

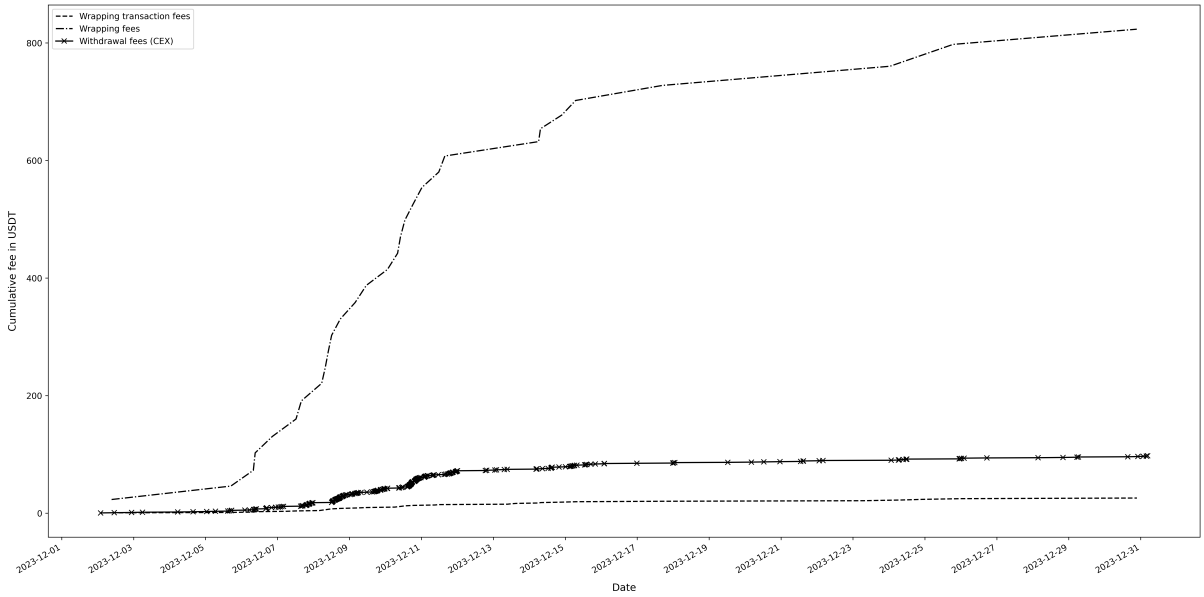
Optimizing spread levels poses a significant challenge for arbitrage traders. To set the profit-maximizing arbitrage spread that yields the highest return on investment, an arbitrage trader has to

trade off frequent execution against a high spread ρ. The question arises whether frequent execution at a relatively low spread level is prioritized over non-frequent execution at a relatively high spread. Ranked by net profit, the top 10% of all arbitrage trades account for 44.55% of the overall profit. For the corresponding two exchanges, it indicates that sufficient liquidity in times of liquidity shortage is as important as a high execution rate.

To determine an appropriate spread ρ, it is essential to systematically log and analyze spread distribution over time. Alternatively, the trader can track the volatility of the arbitrage currencies and increase the spread with higher volatility. Flexible spread management, which allows spread adjustment depending on liquidity distribution, helps to reduce the cost of circulating funds. If the bot is low on liquidity at one exchange, the desired spread level is reduced, which helps to rebalance liquidity automatically so that the travel costs $F_{\text{transaction}}$ and $F_{\text{withdrawal}}$ are reduced.

Another strategy is to study peer arbitrage traders and place orders accordingly. Every interaction on a public blockchain is recorded, allowing the study of the behavior of peer traders, which can help optimize the spread level to be competitive against other arbitrage traders.

Figure 4.3: Cumulative fees dMFP



To evaluate the overall performance of the bot, figure 4.3 shows the costs that occur through the rotation of liquidity. The travel cost C_{travel} incorporates the withdrawal cost $F_{\text{withdrawal}}$ arising whenever DFI is withdrawn from the CEX. A total of 246 withdrawals with an average withdrawal volume of 9118.18 DFI are withdrawn during the time horizon of the experiment. A total of 2,243,071 DFI is withdrawn at a cost of $97.54.

Further, the travel cost C_{travel} includes costs that occur through wrapping activities F_{wrapping}. The wrapping provider of the DeFiChain charges a variable fee of 25 bps on the wrapped volume, along with a fixed fee to cover the Solana transaction cost $F_{\text{transaction}}$ for transferring funds from the wrapping provider to the CEX. A total of 4,870.14 Solana is circulated from the blockchain through the wrapping provider towards the CEX. The average withdrawal volume is 143 Solana, at a total cost of $25.73. The associated total wrapping fees amount to $823.42, which is the highest cost factor after the trading fees. The initial portfolio value amounts to $21,565.5 on November 27, 2023. The total trading profit amounts to $6318.47, and the net profit is $5371.8. The return on investment between November 27, 2023, to December 31, 2023, amounts to 24.9%.

In the following, the paper analyzes the second arbitrage system dD6K, which combines Ethereum and Solana arbitrage bots. Initially, the goal was to operate the Ethereum and Solana bots independently; however, this approach was later adjusted to improve liquidity efficiency. The initial exposure portfolio for the dD6K is 50,000 DFI, 4,200 USDT, 150 SOL, and 20.2 ETH. While two arbitrage bots of the dD6K operate independently, they share common liquidity in DFI and USDT exposure. To maintain a trade volume of $1,500 for each strategy, the exposure position in DFI and USDT is increased by 25,000 DFI and 2,100 USDT. This adjustment ensures that both arbitrage bots do not compete for liquidity simultaneously, thereby preventing potential interference when accessing DFI or USDT resources. As with the dMFP arbitrage system, we selected a trade volume of $1,500 to effectively manage the expected slippage for both arbitrage bots.

The Ethereum and Solana bots perform 996 and 1236 successful arbitrage trades, respectively. The average realized profit per trade is 1.24% ($5.57) for the Solana bot and 1.62% ($10.04) for the Ethereum bot. For all three bots in the two arbitrage systems, the average realized profit exceeds the expected realized profit, which is a result that arises through outlier trades. A total of four trades are unprofitable, leading to an overall loss of $5.85. An unprofitable trade can occur due to price changes or slippage at either exchange. The historical trade data shows that the reason for all four unprofitable trades is due to price changes at the DEX, as the trade output at the DEX for each unprofitable trade was smaller than the CEX trade input.

Figure 4.4: Trade profitability dD6K

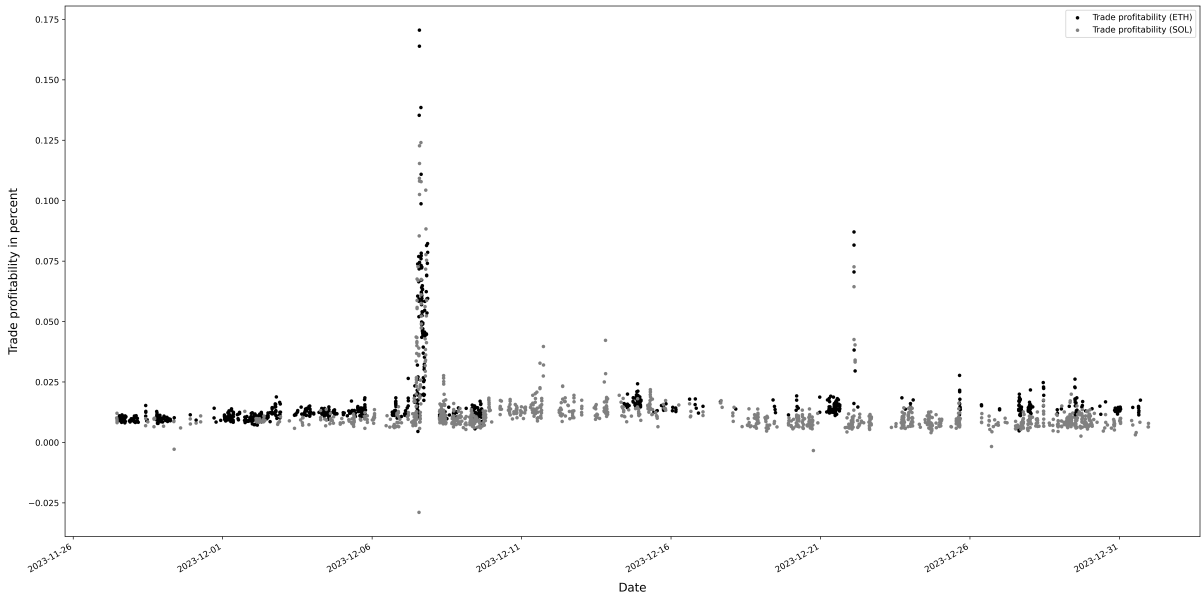


Figure 4.4 illustrates the distribution of realized arbitrage profit in percent for both bots from the dD6K arbitrage system. As with the dMFP trading system, the realized profit incorporates the trading costs C but excludes the cost C_{travel} that occurs whenever liquidity rotates between the exchanges. The trading fees and configuration of both bots are the same as for the bot in the dMFP system, targeting an expected profit of 70 bps. Figure 4.5 shows the cumulative trade profit for the dD6K arbitrage system. It depicts the profit of the Ethereum and Solana bots that achieve a total trading profit of $9999.84 and $6884.52, respectively. For this, it required a total trading volume filled for bid and ask limit orders at the CEX that amounts to $1,060,672. The top 10% of all arbitrage trades account for 51.67% of the overall profit. Figure 4.6 depicts the cumulative travel costs for the dD6K arbitrage system. The system combines two trading bots that have withdrawn liquidity from the CEX to the blockchain whenever DFI

Figure 4.5: Cumulative profit dD6K

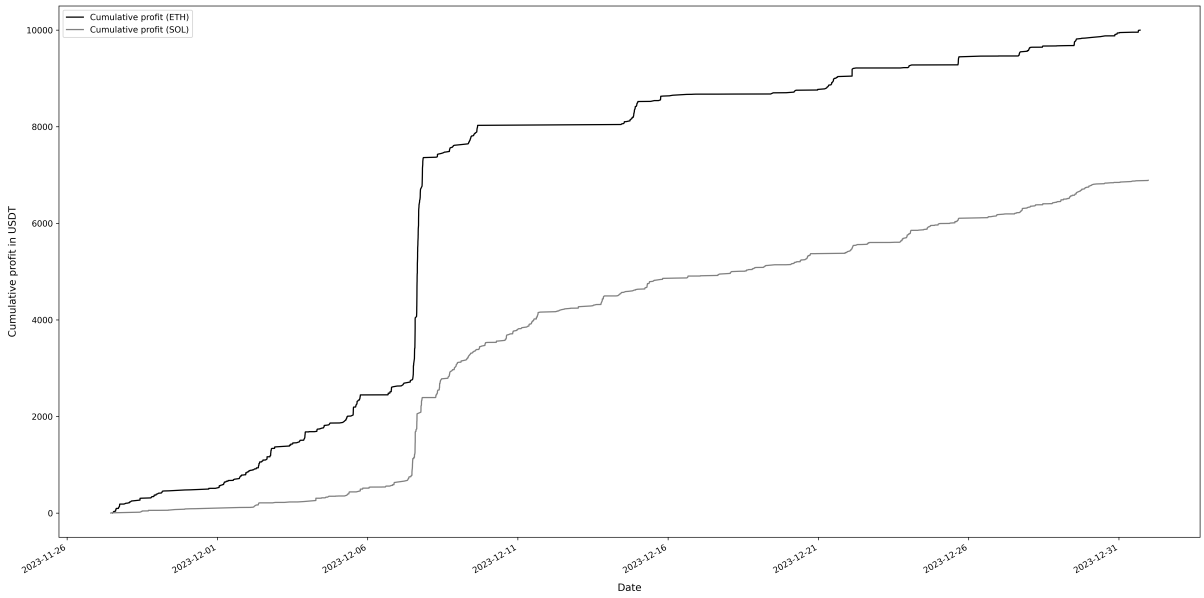
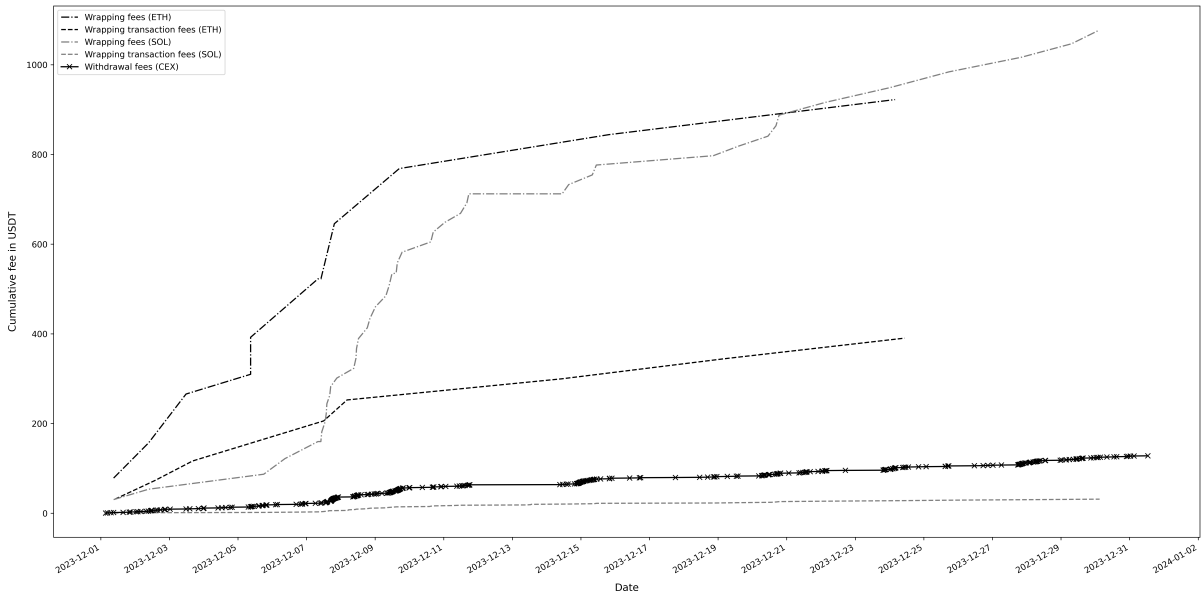


Figure 4.6: Cumulative fees dD6K



liquidity is low. A total of 323 withdrawals are executed at an average volume of 17,141 DFI. A total of $128.07 was paid to withdraw 5,536,804 DFI. The return on investment for the dD6K arbitrage system is 33.35% for the Solana bot and 17.39% for the Ethereum bot.

Table 4.1: Performance summary of arbitrage systems

	dMFP system (SOL)	dD6K system (SOL)	dD6K system (ETH)
Portfolio DFI	25,000 ($6,150)	50,000 ($12,300)	x
Portfolio USDT	2,100 ($2,100)	4,200 ($4,200)	x
Portfolio ETH	x	x	20.2 ($41,694)
Portfolio SOL	225 ($13,316)	150 ($8,877)	x
Total portfolio value	21,565	67,071	-
Trades	1,074	1,236	996
Average realized spread	1.54%	1.24%	1.62%
Average profit	$5.88	$5.57	$10.04
Total trading profit	6,318.47	6,884.52	9,999.84
CEX DFI withdrawals	246	323	-
CEX average volume DFI withdrawal	9,118 DFI	17,141 DFI	-
Total withdrawn amount	2,243,071	5,536,804	-
Total cost	97.54$	128.07$	-
Total wrapping volume	4,870.14 SOL	5,794.71 SOL	164.56 ETH
Average withdrawal	143.24 SOL	137.97 SOL	18.28 ETH
Total transaction fee	25.73$	31.41$	390.42$
Total wrapping fee	823.42$	1077.11$	922.31$
ROI	24.91%	33.35%	17.39%
Net profit	5,371.78$	5,711.97$	8,687.14$

Table 4.1 summarizes the performance of both arbitrage systems. A comparison between the dMFP and dD6K arbitrage systems reveals the economies of scale advantage inherent in an arbitrage system that uses multiple bots. The average withdrawal volume is 17,141 DFI, unlike 9118.18 DFI for the dMFP arbitrage system. The withdrawal cost is a fixed cost, which allows the dD6K arbitrage system to rotate DFI liquidity more cost-efficiently compared to the dMFP arbitrage system. Another advantage that results from the combined portfolio is an increased number of executions. The dMFP arbitrage system utilizes 225 Solana for arbitrage trading, while the dD6K arbitrage system utilizes 150 Solana. However, the dD6K system has 1236 successful arbitrage trades compared to the 1074 arbitrage trades of the dMFP system.

The distribution of realized arbitrage trades for both systems shows that multiple executions are centered on specific dates. In other words, if arbitrage opportunities occur, they often persist and allow multiple executions. The dD6K arbitrage strategy benefits during a price shock in one currency, as the shared liquidity of DFI permits multiple executions until the bot exhausts its DFI liquidity. This explains the higher number of executions for the dD6K system compared to the dMFP system, which holds a larger exposure position in Solana. The total wrapping fees for the dD6K arbitrage system amount to $1077.11 and $31.41 for the wrapping and transaction fee for the Solana bot, and $922.3 and $390.4 for the Ethereum bot.

The main reason for the high number of profits for both arbitrage systems is the price divergence of DFI to SOL and ETH between November 27, and December 31, 2023. While DFI depreciated from $0.246 to $0.159, Solana and Ethereum appreciated from $59.18 to $101.99 and $2064.07 to $2294.34, respectively. The static price of the AMM protocol allowed arbitrage traders to continuously benefit from price discrepancies between the DEX and CEX. As the price divergence comes to the advantage of arbitrage profits, it comes with a loss in the exposure position. The exposure position for dD6K, which has 50,000 DFI, results in a total loss of $4,350.

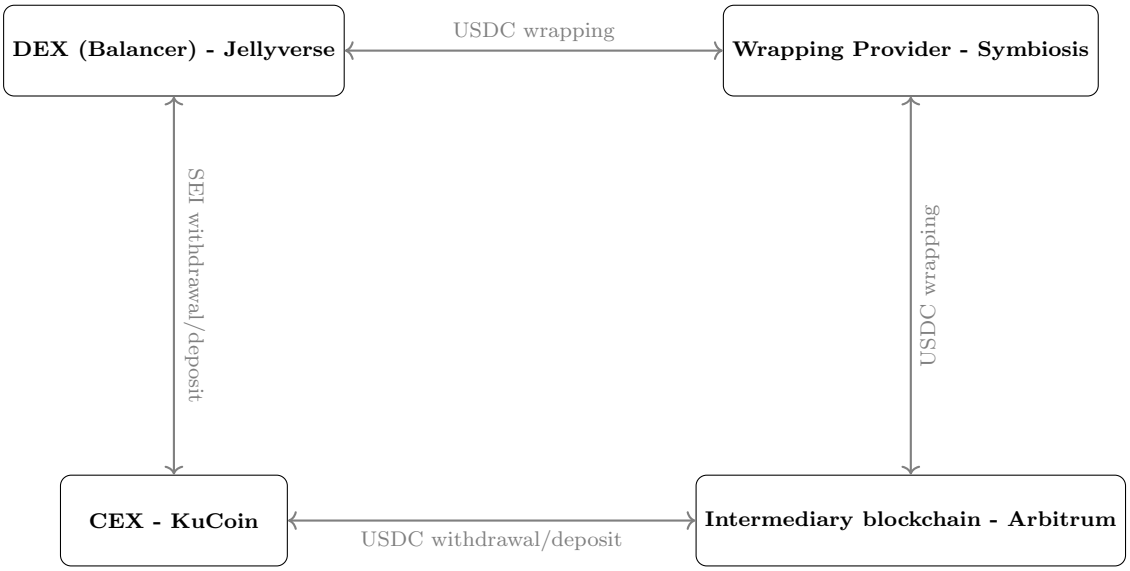
In traditional financial markets, a trader can hedge their exposure position. However, most medium and small-cap cryptocurrencies cannot be shorted, leaving arbitrage traders unable to hedge their exposure risk. As a result, they must account for this risk, leading to higher expected spreads and less capital-efficient markets. To counteract this issue, it is beneficial for blockchains to support the develop-

ment of lending protocols such as AAVE or Compound, which allow for shorting cryptocurrencies. Gogol et al. (2024) indicate that lending protocols allow users to borrow tokens against collateral. Users pay an interest rate on the loan capital, which is dependent on the supply and demand of the corresponding coins in the lending protocol. Arbitrage traders could deposit a stablecoin such as DAI, USDT, or USDC and lend SOL and DFI funds. However, for security reasons, lending protocols only support a limited set of coins for deposits and lending activities. Bartoletti et al. (2021) provide a systematization of knowledge for lending protocols and explain associated risks when interacting with these protocols. They describe circumstances in which users can get liquidated and indicate the interest rate risk that arises through flexible rates of the lending protocol. To reduce the risk of liquidation, users can overcollateralize their position in the smart contract. Whereas this is a risk-reducing approach, it comes with the trade-off of an increased exposure position and reduced return on investment for an arbitrage trader.

5 Robustness check

Following the successful deployment of the arbitrage trading system on DeFiChain, our next objective is to evaluate the algorithm's robustness by determining its adaptability across other blockchain platforms. To select an appropriate blockchain for this testing phase, we prioritize blockchains that exhibit similar characteristics, including low transaction costs, availability of AMM smart contracts with a minimum of $0.5 million in TVL, and the ability to facilitate the circulation of wrapped assets, thereby enabling arbitrage conditions. The SEI blockchain is identified as a suitable environment for testing the algorithm. The SEI blockchain's block time is approximately 400 milliseconds, with a throughput of up to 12,500 transactions per second.[25] SeiScan[26] shows historical transaction fees that are consistently below $0.05, aligning well with the algorithm's requirements. Jellyverse,[27] an AMM protocol based on Balancer model, offers weighted and stable pools with sufficient liquidity, ensuring the bot can operate with sufficient trading volumes.[28] Symbiosis operates as a bridge provider to circulate USDC funds towards the SEI blockchain.[29]

Figure 5.1: SEI and USDC flow



As figure 5.1 shows, USDC liquidity can be withdrawn from Kucoin to the Arbitrum blockchain and

[25] See official website.
[26] See official website.
[27] See official website.
[28] See official website.
[29] See official website.

subsequently bridged to the SEI blockchain. Kucoin remains the CEX as it allows the algorithm to benefit from hidden orders. For trading at Jellyverse and Kucoin, the third trading system 0x2f uses the trading pairs SEI-USDC and SEI-USDT, respectively.[30] To circulate USDC through Symbiosis from Kucoin to the SEI blockchain, we use the CEX trading pair USDC-USDT to convert USDT to USDC. Unlike the triangular arbitrage strategy used on the DeFiChain, we implement a two-point arbitrage strategy on the SEI blockchain, assuming USDT and USDC maintain a stable parity value. The execution of two-point arbitrage has lower trading fees, as it requires only two trades rather than three. Furthermore, the limit order fee on KuCoin for the SEI market is 8 bps, compared to 16 bps for DFI. Given the bot's reduced transaction costs, the spread threshold is configured at 70 bps to enhance execution frequency.

The comparably low TVL (approximately $2,200,000) in the smart contract forces the configuration to operate under a reduced trading volume of $600. This is because the configuration aims to keep the expected slippage low to guarantee profitable arbitrage trades. The 0x2f system operates from September 12, 2024, to October 08, 2024, and uses an exposure position of 12,000 SEI, 2,100 USDT, and 2100 USDC. At the start of the time horizon, prices were: SEI $0.282, USDT $1, and USDC $1. A trade on the Balancer fork, Jellyverse, for the SEI-USDC trading pair charges a fee of $f_{\text{DEX}} = 30$ bps, the limit order fee amounts to $f_{\text{CEX}_{\text{limit}}} = 8$ bps and the spread level is configured as $\rho = 70$ bps. For a trading volume of $600, the expected slippage at the DEX is $s_{\text{DEX}} = 3$–5 bps. Correspondingly, the expected profit for the 0x2f system is 27–29 bps.

Figure 5.2: Trade profitability 0x2f

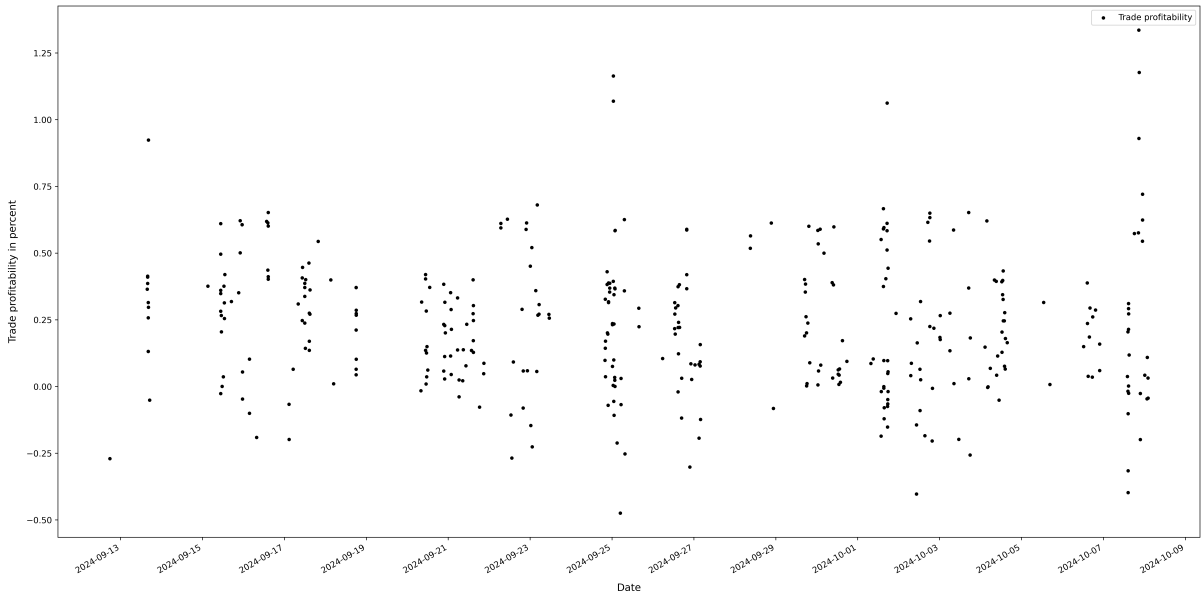


Figure 5.2 depicts the distribution of realized arbitrage trades for the 0x2f system. The average realized arbitrage spread is 23 bps which aligns with the expected value. Out of 411 trades, 319 trades are profitable, and 92 are unprofitable, with the total loss from all unprofitable trades amounting to $47.77. Frequent minor negative trades occur because, instead of wrapping USDC and USDT through the wrapping provider, we aim for auto rebalancing trade. As discussed in subsection 2.2 we apply autorebalancing trades to reduce/mitigate travel costs. To avoid the circulation of funds, which are connected with comparably high costs C_{travel}, autorebalancing allows the reduction of the spread level for either order (bid or ask). If the bot runs out of liquidity on one position, autorebalancing reduces the spread level. Whereas the design of the bot targets a profit of around 27–29 bps, "autorebalancing

[30] The wallet address is 0x98890e37bfE73ED58944706076E37318769EB52f.

Figure 5.3: Cumulative profit 0x2f

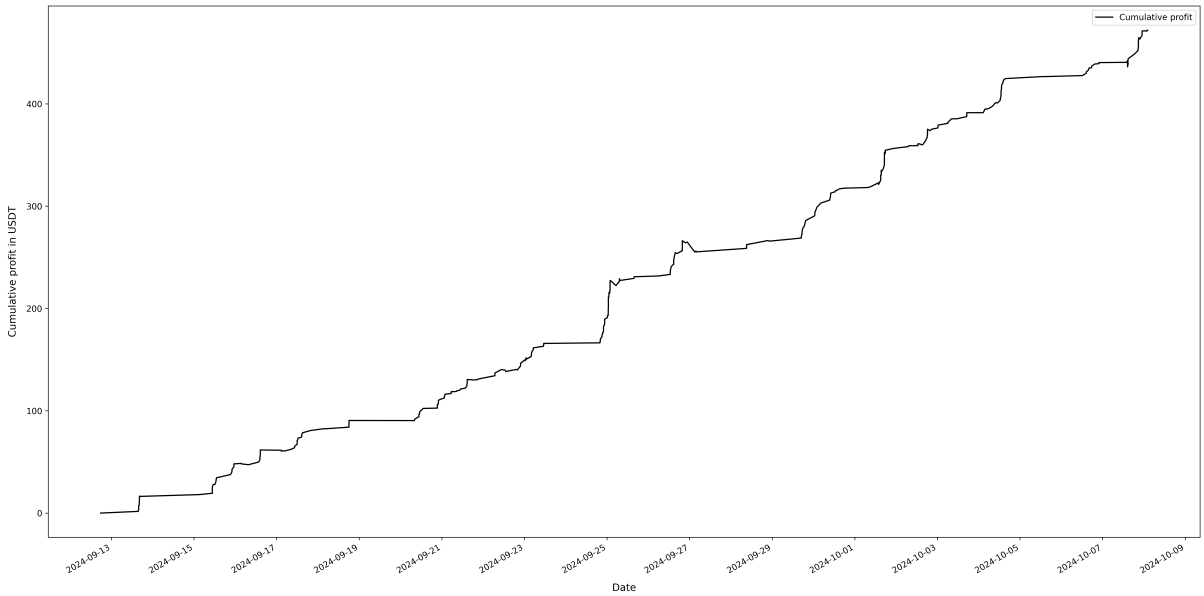
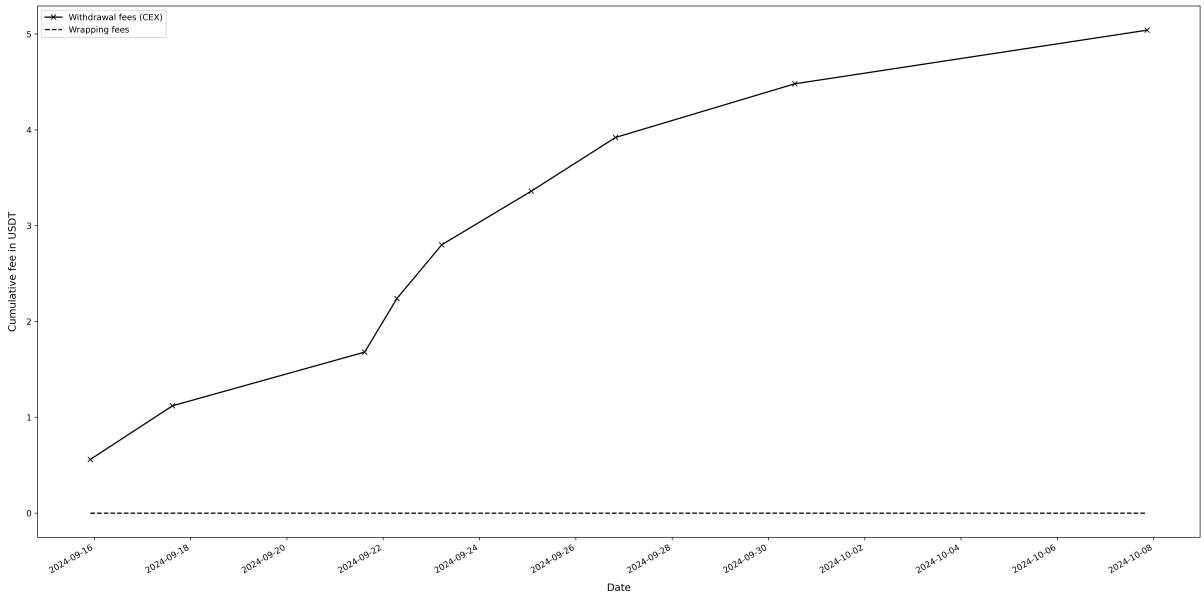


Figure 5.4: Cumulative fees 0x2f



= 30 bps" further reduces the profit for such trades to around 0 bps.

Figure 5.4 depicts the travel fees C_{travel} that arise through the circulation of capital between the two exchanges. A total of 9 withdrawals with an average volume of 5,444 SEI are withdrawn from the CEX to the SEI blockchain, incurring a total withdrawal cost of $F_{\text{withdrawal}} = \$5.04$. The 0x2f trading system rebalanced its exposure position via autorebalancing, eliminating the need for USDC or USDT circulation and thereby avoiding any wrapping fees $F_{\text{wrapping}} = \$0$. However, the avoidance of wrapping fees comes at the cost of negative trades amounting to \$47.77.

Figure 5.3 shows the cumulative profit over the 26-day period for the trading system 0x2f, demonstrating a total profit of \$471.73 and confirming the algorithm's robust performance on a new blockchain. The net profit of the 0x2f trading system is \$466.70, yielding a return on investment of 6.15%. It shows that arbitrage trading in the field of cryptocurrencies can be highly profitable. However, various risk components remain. Arbitrage traders have an exposure risk that cannot be hedged for every coin. Frontrunning attacks and arbitrage competitors can lead to unprofitable trades. Wrapped funds are dependent on the trust of the issuer and can lead to a total exposure loss. Therefore, the risks and cost structure of the arbitrage system should be analyzed to choose a suitable spread that allows building a profitable system.

6 Conclusion

The paper demonstrates the lucrativeness of arbitrage trading between decentralized and centralized cryptocurrency exchanges. Three independent arbitrage trading systems earned a total profit of 19,770.78 USDT for the time horizon from November 27, 2023, to December 31, 2023, and 466.70\$ from September 12, 2024, to October 08, 2024. For all three arbitrage trading systems, the realized arbitrage spreads align with or exceed the expected spreads.

Practical arbitrage differentiates from theoretical backtesting of arbitrage opportunities as it includes all operational costs, liquidity constraints, and direct effects on the markets. Therefore, our economic model includes all operational costs and frictions arbitrage traders exhibit at praxis and documents them empirically. Further, it discusses the trade-offs for various cost adjustments to maximize the profit.

We demonstrate various issues when interacting with smart contracts that need to be studied before setting up an arbitrage system. Unlike high-frequency trading at CEXs, trading at the DEX is not purely dependent on fast executions. We disclose our trading bot's algorithm, enabling readers to replicate it on blockchains with similar characteristics. Arbitrage profits for traders typically arise at the expense of liquidity providers at DEXs, who experience impermanent loss due to the static nature of the liquidity pool design. To maintain liquidity, it is advisable for DEXs to explore improvements to their static pricing mechanism of the protocol. While blockchain innovations, such as DEX aggregators, have enhanced the efficiency of order flow for on-chain activities, there remains significant potential to improve the order flow between DEXs and CEXs to reduce cross-exchange spreads and optimize market efficiency. Smart contract-based solutions that connect both exchanges, as proposed by Jansen (2023), could help to overcome the inefficiency of fragmented capital in the cryptocurrency markets.

References

Bartoletti, M., Chiang, J. H. Y., and Lafuente, A. L. (2021). Sok: Lending pools in decentralized finance. *Financial Cryptography and Data Security. FC 2021 International Workshops: CoDecFin, DeFi, VOTING, and WTSC*, 553–578.

Baum, C., Chiang, J. H.-y., David, B., Frederiksen, T. K., and Gentile, L. (2022). Sok: Mitigation of

front-running in decentralized finance. *International Conference on Financial Cryptography and Data Security*, 250–271.

Brauneis, A., Mestel, R., Riordan, R., and Theissen, E. (2021). How to measure the liquidity of cryptocurrency markets? *Journal of Banking & Finance*, 124:1–26.

Bruzgė, R. and Šapkauskienė, A. (2022). Network analysis on bitcoin arbitrage opportunities. *The North American Journal of Economics and Finance*, 59:1–16.

Bukov, A., Kunz, S., Melnik, M., Alekseev, G., Snow, M., and Shape, X. (2024). Intent-based atomic cross-chain swaps.

Buterin, V. (2014). Ethereum: A next-generation smart contract and decentralized application platform. *White paper*.

Caldarelli, G. (2021). Wrapping trust for interoperability: A preliminary study of wrapped tokens. *Information*, 13(1):1–25.

Crépellière, T., Pelster, M., and Zeisberger, S. (2023). Arbitrage in the market for cryptocurrencies. *Journal of Financial Markets*, 64:1–31.

Daian, P., Goldfeder, S., Kell, T., Li, Y., Zhao, X., Bentov, I., and Juels, . (2019). Flash boys 2.0: Frontrunning, transaction reordering, and consensus instability in decentralized exchanges. *arXiv*, 1–23.

DeFiChain (2019). Defichain ecosystem. *White Paper*.

Garvey, R. and Wu, F. (2010). Speed, distance, and electronic trading: New evidence on why location matters. *Journal of Financial Markets*, 13(6):367–396.

Gogol, K., Killer, C., Schlosser, M., Bocek, T., Stiller, B., and Tessone, C. (2024). Sok: Decentralized finance (defi)–fundamentals, taxonomy and risks. *arXiv*, 1–20.

Griffin, J. M. and Shams, A. (2020). Is bitcoin really untethered? *The Journal of Finance*, 75(4):1913–1964.

Jansen, M. (2023). Secure arbitrage trades between centralized and decentralized exchanges. *Proceedings of the 2nd Blockchain and Cryptocurrency Conference*.

Jiao, T., Xu, Z., Qi, M., Wen, S., Xiang, Y., and Nan, G. (2024). A survey of ethereum smart contract security: Attacks and detection. *Distributed Ledger Technologies: Research and Practice*, 3(3):1–28.

Jin, H., Dai, X., and Xiao, J. (2018). Towards a novel architecture for enabling interoperability amongst multiple blockchains. *IEEE 38th International Conference on Distributed Computing Systems*, 1203–1211.

JupiterChain (2018). Smart consentable data exchange. *White Paper*.

KCS.Foundation (2022). A blockchain-based value self-circulation ecosystem. *White Paper*.

Loesch, S., Hindman, N., Richardson, M. B., and Welch, N. (2021). Impermanent loss in uniswap v3. *arXiv*, 1–43.

Makarov, I. and Schoar, A. (2020). Trading and arbitrage in cryptocurrency markets. *Journal of Financial Economics*, 135(2):293–319.

Martinell, F. and Mushegian, N. (2019). A non-custodial portfolio manager, liquidity provider, and price sensor. *White Paper*.

Păuna, C. (2018). Arbitrage trading systems for cryptocurrencies: Design principles and server architecture. *Informatica Economica*, 22(2):35–42.

Schleifer, A. and Vishny, R. W. (1997). The limits of arbitrage. *The Journal of Finance*, 52(1):35–55.

SeiLabs (2024). Sei: A high-performance layer 1 blockchain optimized for trading. *White Paper*.

Uniswap (2020). Uniswap v2 core: White paper. *White Paper*.

Xu, J., Paruch, K., Cousaert, S., and Feng, Y. (2023). Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Computing Surveys*, 11(55):1–50.

Yakovenko, A. (2017). Solana: A new architecture for a high-performance blockchain. *White Paper*.